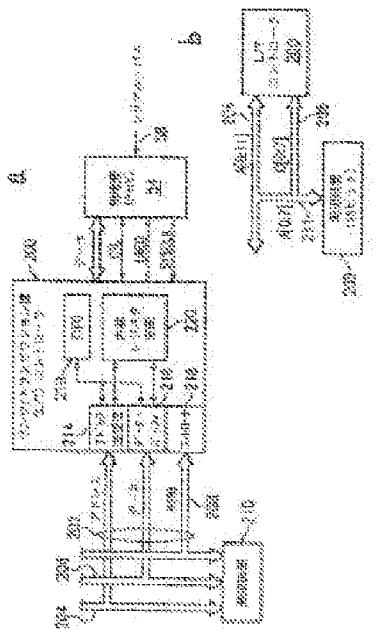
LINK AND TRANSACTION LAYER CONTROLLER WITH INTEGRATED MICROCONTROLLER EMULATOR

Publication number: JP2000188626 (A)
Publication date: 2000-07-04
Inventor(s): GUGEL BOB; MERRYL NEWMAN; BURKE HENEHAN +
Applicant(s): TEXAS INSTRUMENTS INC +
Classification:
- international: ***G06F13/38; G06F13/42; H04L12/40; H04L29/10;*** **G06F13/38; G06F13/42; H04L12/40; H04L29/10;** (IPC1-7): G06F13/38; G06F13/42; H04L12/40; H04L29/10
- European:
Application number: JP19990327311 19991013
Priority number(s): US19980103937P 19981013

Abstract of **JP 2000188626 (A)**

PROBLEM TO BE SOLVED: To interface data reception from an IEEE 1394 serial bus with a peripheral unit in a host system, without the need for a separate microcontroller. SOLUTION: A physical layer 54 that extracts data and a link transaction layer controller 200, that interfaces data from the physical layer 54 with a host system, are provided between an IEEE 1394 serial bus 58 and the host system. The host system consists of a peripheral unit 210 and a host system bus 202. The link transaction layer controller 200 is made operable so as to emulate a microcontroller function and addresses and data are transferred to the peripheral unit 210. That is, a remote node can access the peripheral unit 210 by transmitting the address and data information. Furthermore, the peripheral unit 210 transmits the address and data information to the link transaction layer controller 200, and the controller processes the received address and data and transmits them to a remote node.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2000－188626
（P2000－188626A）
(43)公開日　平成12年７月４日(2000.7.4)

(51)Int.Cl.[7]		識別記号	Ｆ Ｉ		テーマコード（参考）
Ｈ０４Ｌ	29/10		Ｈ０４Ｌ	13/00	３０９Ｃ
Ｇ０６Ｆ	13/38	３５０	Ｇ０６Ｆ	13/38	３５０
	13/42	３２０		13/42	３２０Ｃ
Ｈ０４Ｌ	12/40		Ｈ０４Ｌ	11/00	３２０

審査請求　未請求　請求項の数16　ＯＬ　外国語出願　（全 71 頁）

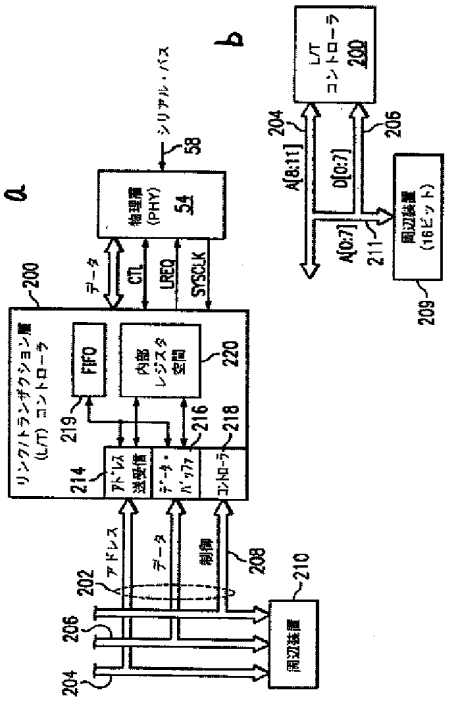
(21)出願番号　特願平11－327311
(22)出願日　平成11年10月13日(1999.10.13)
(31)優先権主張番号　１０３９３７
(32)優先日　平成10年10月13日(1998.10.13)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　590000879
テキサス　インスツルメンツ　インコーポレイテツド
アメリカ合衆国テキサス州ダラス，ノース　セントラルエクスプレスウエイ　13500
(72)発明者　ボブ　グゲル
アメリカ合衆国，テキサス，ガーランド，オフェラン　レーン　810
(72)発明者　メリル　ニューマン
アメリカ合衆国，テキサス，キャロルトン，レミントン　ドライブ　3737
(74)代理人　100066692
弁理士　浅村　皓　（外３名）

最終頁に続く

(54)【発明の名称】　一体のマイクロコントローラ・エミュレータを有するリンク／トランザクション層コントローラ

(57)【要約】
【課題】　ＩＥＥＥ１３９４シリアル・バスからのデータ受信において、別個のマイクロコントローラを必要とせずにホスト・システム内の周辺装置にインタフェースすること。
【解決手段】　ＩＥＥＥ１３９４シリアル・バス５８とホスト・システムの間に、データを抽出する物理層５４とデータを物理層からホスト・システムにインタフェースするリンク・トランザクション層コントローラ２００が設けられる。ホスト・システムは周辺装置２１０とホスト・システム・バス２０２から成る。リンク・トランザクション層コントローラ２００はマイクロコントローラ機能をエミュレートするよう動作可能で、アドレスはデータと共に周辺装置２１０に転送される。即ち、遠隔ノードはアドレスおよびデータ情報を送信することによって周辺装置２１０にアクセスできる。また、アドレスおよびデータ情報を周辺装置２１０からリンク・トランザクション層コントローラ２００に送信することができ、該コントローラは受信したアドレスおよびデータを処理してそれを遠隔ノードに送信する。



1

【特許請求の範囲】

【請求項１】　シリアル・バスとホスト・システム間にインタフェースするため、および、遠隔ノードにより前記シリアル・バス上に置かれた情報を前記シリアル・バスから受信しかつ該受信した情報を前記ホスト・システムに転送し、また、前記ホスト・システムから情報を受信しかつ該受信した情報を前記遠隔ノードによる受信のため前記シリアル・バスに転送するためにローカル・ノード上に設けられたシリアル・バス・インタフェースであって、前記遠隔ノードにより生成されたデータを前記シリアル・バスから受信するためのデータ受信器と、データを前記遠隔ノードでの受信のため前記シリアル・バスに送信するためのデータ送信器と、少なくとも１つのレジスタがそこに受信データを記憶のため前記遠隔ノードによりアドレス可能である複数のレジスタを備えていて、前記データ受信器は読み出し動作の間受信したデータを前記少なくとも１つのレジスタに記憶するよう動作可能であり、また、前記データ送信器は書き込み動作の間データを前記シリアル・バスに送信するようになっており、さらに、前記ホスト・システム上のホスト・バスに直接インタフェースするためのホスト・バス・インタフェースを備えていて、該ホスト・バス・インタフェースは、前記少なくとも１つのレジスタに記憶されたデータを、書き込み動作の間、該データが受信されて前記少なくとも１つのレジスタに記憶されるとき、前記ホスト・バスに転送し、また、読み出し動作の間データを前記ホスト・バスから取り出すようになっているシリアル・バス・インタフェース。

【請求項２】　請求項１記載のシリアル・バス・インタフェースにおいて、シリアル・バス・インタフェースが標準レジスタ空間を含んでいて、前記少なくとも１つのレジスタは該標準レジスタ空間の一部を占有しているシリアル・バス・インタフェース。

【請求項３】　請求項１記載のシリアル・バス・インタフェースにおいて、前記複数のレジスタのうちの選ばれたレジスタは標準バス・インタフェース情報の記憶に専用に供されていて、遠隔ノードは前記標準バス・インタフェース情報に関連付けられた前記複数のレジスタを直接アドレスしてそこから該情報にアクセスでき、また、前記データ受信器は前記複数のレジスタのうちの１つへのアクセス要求を認識するよう動作可能でかつ前記データ送信器はアドレスされるときその内容を送信するよう動作可能であるシリアル・バス・インタフェース。

【請求項４】　請求項３記載のシリアル・バス・インタフェースにおいて、前記複数のレジスタのうちの選ばれたレジスタはコンフィギュレーション・レジスタから成っていて、該コンフィギュレーション・レジスタは前記シリアル・バスの動作を定めるコンフィギュレーション情報のために用いられ、それにより、遠隔ノードは前記コンフィギュレーション・レジスタの１つにアクセスすることによって前記シリアル・バスの動作をプログラムできるようになっているシリアル・バス・インタフェース。

2

【請求項５】　請求項１記載のシリアル・バス・インタフェースにおいて、前記データ受信器によって受信されるデータおよび前記データ送信器によって送信されるデータはデータ・パケットであり、該データ・パケットは前記シリアル・バス上の送信ノードを識別するのに必要な情報と、該データ・パケットの内容と、データ・パケットを受信するよう指定された遠隔ノードを識別する情報を含んでいるシリアル・バス・インタフェース。

【請求項６】　請求項５記載のシリアル・バス・インタフェースにおいて、各データ受信または送信動作に先行して前記遠隔ノードからの書き込み要求または読み出し要求のデータ要求があり、該要求は前記受信されたデータ・パケット内に含まれており、書き込み要求に関連した前記受信データ・パケットは前記少なくとも１つのレジスタに記憶されたそれに関連したデータを含んでいて、前記ホスト・インタフェースはこの書き込み要求を認識して前記少なくとも１つのレジスタに記憶された前記データを前記ホスト・バスに転送し、読み出し要求では、前記ホスト・インタフェースはこの読み出し要求を認識して該データに前記データ送信器を持つ遠隔ノードへの転送のため前記ホスト・システムからアクセスするシリアル・バス・インタフェース。

【請求項７】　請求項６記載のシリアル・バス・インタフェースにおいて、前記遠隔ノードからの前記書き込み要求は前記少なくとも１つのレジスタに記憶のためのアドレス情報とデータ情報の両方を含んでおり、前記ホスト・インタフェースはアドレス・バスとデータ・バスを持つ前記ホスト・バスにこれらアドレス情報とデータ情報の両方を送信するよう動作可能であるシリアル・バス・インタフェース。

【請求項８】　請求項６記載のシリアル・バス・インタフェースにおいて、前記遠隔ノードからの前記読み出し要求は前記少なくとも１つのレジスタに記憶されているアドレスを含んでおり、前記ホスト・インタフェースは、読み出し要求を認識するとアドレス・バスとデータ・バスを持つ前記ホスト・バスに該アドレスを送信するよう動作可能であり、前記データ送信器による前記遠隔ノードへの送信のため該データ・バスからデータを取り出すシリアル・バス・インタフェース。

【請求項９】　シリアル・バスとローカル・ノード上のホスト・システム間にインタフェースするため、および、遠隔ノードにより前記シリアル・バス上に置かれた情報を前記シリアル・バスから受信しかつ該受信した情報を前記ホスト・システムに転送し、また、前記ホスト・システムから情報を受信しかつ該受信した情報を前記遠隔ノードによる受信のため前記シリアル・バスに転送するための方法であって、前記遠隔ノードにより生成さ

れたデータを前記シリアル・バスから受信するステップと、データを前記遠隔ノードでの受信のため前記シリアル・バスに送信するステップと、少なくとも１つのレジスタがそこに受信データを記憶のため前記遠隔ノードによりアドレス可能である複数のレジスタを設けるステップを含み、前記受信ステップは読み出し動作の間受信したデータを前記少なくとも１つのレジスタに記憶するよう動作可能であり、また、前記送信ステップは書き込み動作の間データを前記シリアル・バスに送信するようになっており、さらに、前記ホスト・システム上のホスト・バスに直接インタフェースするためのホスト・バス・インタフェースが設けられていて、該ホスト・バス・インタフェースは、前記少なくとも１つのレジスタに記憶されたデータを、書き込み動作の間、該データが受信されて前記少なくとも１つのレジスタに記憶されるとき、前記ホスト・バスに転送し、また、読み出し動作の間データを前記ホスト・バスから取り出すようになっている方法。

【請求項１０】　請求項９記載の方法において、シリアル・バス・インタフェースが標準レジスタ空間を含んでいて、前記少なくとも１つのレジスタは該標準レジスタ空間の一部を占有している方法。

【請求項１１】　請求項９記載の方法において、前記複数のレジスタのうちの選ばれたレジスタは標準バス・インタフェース情報の記憶に専用に供されていて、該標準バス・インタフェース情報に関連付けられた前記複数のレジスタを遠隔ノードによって直接アドレスしてそこから該情報にアクセスするステップをさらに含み、前記受信ステップは前記複数のレジスタのうちの１つへのアクセス要求を認識するよう動作可能であり、前記送信ステップはアドレスされるときその内容を送信するよう動作可能である方法。

【請求項１２】　請求項１１記載の方法において、前記複数のレジスタのうちの選ばれたレジスタはコンフィギュレーション・レジスタから成っていて、該コンフィギュレーション・レジスタは前記シリアル・バスの動作を定めるコンフィギュレーション情報のために用いられ、それにより、遠隔ノードは前記コンフィギュレーション・レジスタの１つにアクセスすることによって前記シリアル・バスの動作をプログラムできるようになっている方法。

【請求項１３】　請求項９記載の方法において、前記受信ステップで受信されるデータおよび前記送信ステップで送信されるデータはデータ・パケットであり、該データ・パケットは前記シリアル・バス上の送信ノードを識別するのに必要な情報と、該データ・パケットの内容と、データ・パケットを受信するよう指定された遠隔ノードを識別する情報を含んでいる方法。

【請求項１４】　請求項１３記載の方法において、各データ受信または送信動作に先行して前記遠隔ノードからの書き込み要求または読み出し要求のデータ要求があり、該要求は前記受信されたデータ・パケット内に含まれており、書き込み要求に関連した前記受信データ・パケットは前記少なくとも１つのレジスタに記憶されたそれに関連したデータを含んでいて、前記ホスト・インタフェースはこの書き込み要求を認識して前記少なくとも１つのレジスタに記憶された前記データを前記ホスト・バスに転送し、読み出し要求では、前記ホスト・インタフェースはこの読み出し要求を認識して該データに前記送信ステップでの遠隔ノードへの転送のため前記ホスト・システムからアクセスする方法。

【請求項１５】　請求項１４記載の方法において、前記遠隔ノードからの前記書き込み要求は前記少なくとも１つのレジスタに記憶のためのアドレス情報とデータ情報の両方を含んでおり、前記ホスト・インタフェースはアドレス・バスとデータ・バスを持つ前記ホスト・バスにこれらアドレス情報とデータ情報の両方を送信するよう動作可能である方法。

【請求項１６】　請求項１４記載の方法において、前記遠隔ノードからの前記読み出し要求は前記少なくとも１つのレジスタに記憶されているアドレスを含んでおり、前記ホスト・インタフェースは、読み出し要求を認識するとアドレス・バスとデータ・バスを持つ前記ホスト・バスに該アドレスを送信するよう動作可能であり、前記データ送信器による前記遠隔ノードへの送信のため該データ・バスからデータを取り出す方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は一般にＩＥＥＥ１３９４タイプのシリアル・バスからデータを受信することに関し、特に、別個のマイクロコントローラを必要とせずに、そのデータをホスト・システム内の周辺ユニットにインタフェースさせることに関する。

【０００２】

【従来の技術】ＩＥＥＥは、物理接続と呼ばれる点間リンクで接続された論理ノードのネットワークを含む高性能シリアル・バス・ケーブル環境用の新しい標準を、ＩＥＥＥ１３９４の下で許可した。この物理接続は各ノード上のポートとそれらの間に設けられたケーブルから成る。１つのノードは複数のポートを持つことができ、これにより、分岐したマルチホップ相互接続が可能になる。このトポロジーへの制限はアービトレーション（裁定）プロトコルに必要な一定のラウンド・トリップ（往復）時間要件によって設定される。バス・リセット後に設定されるデフォルト・タイミングは、各ホップ当たり４．５メートルで合計７２メートルに対して１６ケーブルホップが適当である。単一のバス上にサポートされるノードの最大数は６３である。

【０００３】ノードがＩＥＥＥ１３９４シリアル・バスに追加されたりそこから削除されるときはいつも、バス

5

・リセットが生じ、全てのノードを知られた状態にする。バス・リセットの後、ツリー識別（ＩＤ）処理によって一般のネットワーク・トポロジーをツリーに変換する。ツリーでは、１つのノードがルートで指定される。全ての物理接続は親か子あるいは非接続のいずれかでラベルされる。非接続ポートは「オフ」とラベルされ、ほかの如何なるものとも関係しない。ツリーは非循環でなければならず、ループは許されない。そうでないと、ツリー識別処理は終了しないことになる。

【０００４】１３９４ケーブル環境は毎秒９８．３０４メガビット、１９６．６０８メガビットおよび３９３．２１６メガビットの複数のデータ速度をサポートする。最低速度はベース速度として知られており、それより高いデータ速度をサポートするポートは全て低いデータ速度もサポートしなければならない。ベース速度より高いデータ速度が可能なノードは、速度シグナリング位相の間その取り付けポートを介してそのピア（同位ノード）と速度情報を交換する。もしピア・ノードが高速度データを受信できないなら、データはその経路を下って伝搬しないことになる。データは高いデータ速度をサポートする経路を下って伝搬するだけである。

【０００５】データ・パケット送信の間、そのソース・ノードは速度符号、フォーマットおよびトランザクション符号、ソースおよび宛て先ノードのアドレス、およびデータをパケット形式で送る。このパケット内の宛て先フィールドは、それが送信データの受信者であるかどうかを決定するため、各ノードのリンク層によって利用される。データ・パケットの伝送可能な最大速度はバス・トポロジーとバス上のノードによってサポートされるデータ伝送速度とに依存する。データ・パケットが送られる最適速度を決定するには、送信および受信ノードの最大サポート速度ばかりでなくこれら送信および受信ノード間に接続された任意のノードの最大速度を決めなければならない。データ伝送のための最適速度は、データ・パケットの伝送に関係するよう要求されたノードの全てによってサポートされる最高速度に等しい。

【０００６】

【発明が解決しようとする課題】ＩＥＥＥ１３９４バスは、典型的には、バスから情報を抽出するための物理層とバスから抽出したデータをホスト・システムにインタフェースするためのリンク／トランザクション層を必要とする。該ホスト・システムは、典型的には、ホスト・バスとＣＰＵを含む。このＣＰＵには、一般に、バスからのデータが記憶されるＦＩＦＯからデータを抽出する仕事を与えられる。このデータは、ＣＰＵによって取り出された後、適当な周辺ユニットに送信されるか、あるいはＣＰＵによる種々の処理動作に利用される。ＣＰＵはまた、情報をシリアル・バスに送ることができる。これは、先ずＦＩＦＯに情報を記憶させ、次にリンク／トランザクション層に命令を与えてＦＩＦＯからその情報を取り出してそれをシリアル・バスに送信することにより行われる。しかしながら、別個のＣＰＵまたはマイクロコントローラが必要であるという事実は、ディジタル・マイクロホン、ステレオ受信器および送信器等の小型の応用に関してＩＥＥＥ１３９４バスの魅力を減少させている。

【０００７】

【課題を解決するための手段】ここに開示し特許請求の範囲に記載した本発明は、シリアル・バスとホスト・システム間にインタフェースするため、および、遠隔ノードにより前記シリアル・バス上に置かれた情報を前記シリアル・バスから受信しかつ該受信した情報を前記ホスト・システムに転送し、また、前記ホスト・システムから情報を受信しかつ該受信した情報を前記遠隔ノードによる受信のため前記シリアル・バスに転送するためにローカル・ノード上に設けられたシリアル・バス・インタフェースである。該インタフェースは、前記遠隔ノードにより生成されたデータを前記シリアル・バスから受信するためのデータ受信器と、データを前記遠隔ノードでの受信のため前記シリアル・バスに送信するためのデータ送信器を含んでいる。また、複数のレジスタが設けられていて、その少なくとも１つはそこに受信データを記憶のため前記遠隔ノードによりアドレス可能である。前記データ受信器は読み出し動作の間受信したデータを前記少なくとも１つのレジスタに記憶するよう動作可能であり、また、前記データ送信器は書き込み動作の間データを前記シリアル・バスに送信するよう動作可能である。ホスト・バス・インタフェースが前記ホスト・システム上のホスト・バスに直接インタフェースするために設けられていて、該ホスト・バス・インタフェースは、前記少なくとも１つのレジスタに記憶されたデータを、書き込み動作の間、該データが受信されて前記少なくとも１つのレジスタに記憶されるとき、前記ホスト・バスに転送するよう動作可能である。該ホスト・バス・インタフェースはまた、読み出し動作の間データを前記ホスト・バスから取り出すよう動作可能である。

【０００８】本発明に他の観点によれば、シリアル・バス・インタフェースは標準レジスタ空間を含んでいて、前記少なくとも１つのレジスタは該標準レジスタ空間の一部を占有している。また、前記複数のレジスタのうちの選ばれたレジスタは標準バス・インタフェース情報の記憶に専用に供されていて、遠隔ノードはこの情報に直接アクセスできる。前記データ受信器は前記複数のレジスタのうちの１つへのアクセス要求を認識するよう動作可能であり、その後、前記データ送信器はアドレスされるときその内容を送信するよう動作可能である。また、前記複数のレジスタのうちの選ばれたレジスタは前記シリアル・バスの動作を定めるコンフィギュレーション（環境設定）情報を記憶するためのコンフィギュレーションレジスタを含んでいる。これにより、遠隔ノードは

6

前記コンフィギュレーション・レジスタの１つにアクセスすることによって前記シリアル・バスの動作をプログラムできる。

【０００９】本発明に別の観点によれば、データはデータ・パケットで受信され、データ・パケットで送信される。これらのデータ・パケットは、前記シリアル・バス上の送信ノードを識別するのに必要な情報および該データ・パケットの内容を、データ・パケットを受信するよう指定された遠隔ノードを識別する情報に加えて含んでいる。また、各データ受信または送信動作に先行して前記遠隔ノードからの書き込み要求または読み出し要求のデータ要求があり、該要求は前記受信されたデータ・パケット内に含まれている。書き込み要求に関連した前記受信データ・パケットは書き込み要求の受信時に前記少なくとも１つのレジスタに記憶されたそれに関連したデータを含んでいる。前記ホスト・インタフェースはこの書き込み要求を認識し、前記少なくとも１つのレジスタに記憶された前記データを前記ホスト・バスに転送する。読み出し要求の間、前記ホスト・インタフェースはこの読み出し要求を認識し、該データに前記データ送信器を持つ遠隔ノードへの転送のため前記ホスト・システムからアクセスする。

【００１０】

【実施例】図１を参照すると、ＩＥＥＥ１３９４シリアル・バスとして定義されたシリアル・バス・アーキテクチャを用いたシステムのブロック図が示されている。これは、「高性能シリアル・バスのためのＩＥＥＥ標準」（ＩＥＥＥ　Ｓｔａｎｄａｒｄ　ｆｏｒ　ａ　Ｈｉｇｈ－Ｐｅｒｆｏｒｍａｎｃｅ　Ｓｅｒｉａｌ　Ｂｕｓ）、ＩＥＥＥ　ＳＴＤ１３９４－１９９５に定められており、これを引用により援用してここに記載に代える。モジュール１０は、ＣＰＵ１２、メモリ１４、入出力（Ｉ／Ｏ）１６およびＣＰＵ１８を含むものとして示されている。ＣＰＵ１２、メモリ１４、入出力（Ｉ／Ｏ）１６およびＣＰＵ１８は全てシステム内のユニットである。ユニット１２－１８の各々は、モジュール１０に固有のシステム・バスであるパラレル・バス２０にインタフェースしている。加えて、ユニット１２－１８の各々は「バックプレーン」と称されるシリアル・バス２２にインタフェースしている。ＩＥＥＥ１３９４標準に従って動作するシリアル・バス２２は、システム外でブリッジ２４にインタフェースしている。ブリッジ２４とモジュール１０はそれぞれシリアル・バス上の論理ノードを構成する。一般に、シリアル・バス・アーキテクチャはアドレス可能なエンティティである論理ノードによって定義される。これらの論理ノードはそれぞれ独立に再設定および識別でき、１つ以上のノードが単一のモジュール上に存在し得るし、また単一のノード内に１つ以上のユニットが存在し得る。それゆえ、ノードは論理アドレッシングの概念であり、そこでは、モジュールは、物理インタフェースを共有する１つ以上のノードから成り得る物理装置である。単一のノードのアドレス空間は１つ以上のユニットに直接マッピングすることができる。ユニットはディスク・コントローラ、メモリ、ＣＰＵ等の論理エンティティであり得る。所与のユニット内には、独立した制御レジスタを介してアクセス可能かあるいはダイレクト・メモリ・アクセス（ＤＭＡ）コマンド列で一意的にアドレス可能な複数のサブ・ユニットが存在し得る。

【００１１】図１をさらに参照すると、バックプレーン・シリアル・バス２２を用いたモジュール１０内の１つの環境（「バックプレーン環境」と称される）と「ケーブル環境」と称される他の１つの環境から成る２つの環境があるのがみられる。ケーブル環境にインタフェースするノードはそれに関連付けられた「ポート」を有している。ブリッジ・ノード２４はそのようなノードであり、一方の側ではバックプレーン・シリアル・バス２２にそして他方の側ではケーブル２６にインタフェースしている。ケーブル２６は単一のＩ／Ｏノード２８にその中の１つのポートを介してインタフェースしている。Ｉ／Ｏノード２８は他に２つのポートを有していて、その１つはケーブル・シリアル・バス３０を介してブリッジ・ノード３２に接続されている。ブリッジ・ノード３２は、モジュールである他のシステム３４にインタフェースしている点でブリッジ・ノード２４に類似している。システム３４は、システム１０と実質的に同一であってもよいし、バックプレーンを用いた任意の別のタイプのシステムであってもよい。Ｉ／Ｏノード２８の第３のポートはケーブル・シリアル・バス３６を介してＩ／Ｏノード３８の１つのポートにインタフェースしており、Ｉ／Ｏノード３８の他のポートはケーブル・シリアル・バス４０を介してＩ／Ｏノード４２にインタフェースしている。

【００１２】一般に、ケーブル環境は有限の分岐と拡張を有する非循環ネットワークを提供する物理的なトポロジーである。その媒体は信号用の２つの導体対と電源および接地用の１つの対とから成り、これらは異なるノード上のポートを接続する。各ポートはターミネータ、送受信器および簡単なロジック（論理）から成る。ケーブルとポートはノード間のバス中継器として働き、単一の論理バスをシュミレートする。これに対して、バックプレーン環境はマルチドロップ（分岐）バスから成る。これはモジュール内のバックプレーンの長さを走行する２つの単一終端導体から成る。バスに沿って分布したコネクタはノードがバスに「プラグ・イン」するのを可能にする。このシステムは全てのノードにバスをアサートするのを許すワイヤードＯＲ論理を利用する。

【００１３】次に図２を参照すると、シリアル・バス・プロトコルのブロック図が示されている。シリアル・バス・プロトコルは、トランザクション層５０、リンク層

５２および「ＰＨＹ」とラベルした物理層５４を含む３つの積層から成っている。トランザクション層５０は、ＣＲＳアーキテクチャ（制御および状態レジスタ）をサポートするよう求められたバス・トランザクションを実行する完全な応答・応答プロトコルを定める。これは読み出し、書き込みおよびロックの動作を提供する。リンク層５２はトランザクション層５０に肯定応答データグラム（要求の確認を伴う一方向のデータ転送）サービスを提供する。それはパケットの送信および受信のためにアドレスの指定、データのチェックおよびデータのフレーミングを行う。リンク層５２はまた、タイミングおよび同期化のために用いられる「サイクル」信号の発生を含む等時性データ伝送サービスをアプリケーションに対して直接提供する。１つのリンク層転送は「サブ・アクション」と呼ばれる。

【００１４】物理層５４は３つの主要な機能を提供する。物理層５４はリンク層５２によって用いられる論理記号を異なるシリアル・バス媒体上の電気信号に変換する。物理層５４は、アービトレーション・サービスを提供することによって、一時に１つのノードだけがバス上にデータを送信していることを保証する。物理層５４はまた、シリアル・バスに対する機械的インタフェースを定める。ケーブルおよびバックボーン環境の各環境毎に異なる物理層がある。ケーブル物理層は、データ再同期化やリピートサービスおよび自動バス初期化を行う。

【００１５】これら３つの層に加えて、ノードを制御したりバス資源を管理するのに必要な基本的な制御機能や標準的なＣＳＲを提供するシリアル・バス管理ブロック（マネージャ）５６が設けられている。このブロック５６は多数の要素から成っていて、それらは、バス全体に亙って管理責任を行使する単一のノードでのみ活性のバス・マネージャ要素や、ノード・コントローラ要素を含む。さらに、データを他の等時性資源に割り当てるのに必要なサービスを集中化する等時性資源マネージャを含んでいる。等時性資源とは、最小持続時間と同じかその整数倍の持続時間のいずれかを有する連続した有効インスタンス間の時間間隔を持つ時間尺度または信号の特性を有する資源のことである。本発明の目的のために、シリアル・バス５８とリンク層５２にインタフェースする物理層５４は受信バッファ（図示されていない）にインタフェースするだろう。

【００１６】次に図３を参照すると、物理層５４とリンク層５２間のインタフェースのブロック図が示されている。物理層５４はシリアル・バス５８にインタフェースし、そこからデータを受信するよう動作可能である。データ（Ｄ）は8ビット双方向データ・バス６０を介してリンク層５２へあるいはリンク層５２から伝送される。２つの制御ビット（ＣＴＬ）が制御バス６２上を物理層５４とリンク層５２の間を伝送される。リンク要求（ＬＲＥＱ）は要求線６４を介してリンク層５２から物理層５４へ転送されるが、ここでシステム・クロック信号ＳＣＬＫが物理層５４からリンク層５２へ転送され、物理層５４はこのクロックを復元する。

【００１７】以下、データ速度を９８．３０４メガビット／秒の倍数で言及する。ケーブル環境でのＩＥＥＥ１３９４に規定されたインタフェースは１００メガビット／秒、２００メガビット／秒および４００メガビット／秒のデータ速度をサポートする。バックプレーン環境は２５メガビット／秒および５０メガビット／秒をサポートする。これらの速度は符号化スキームと無関係な実際の「ビット」レートである。冗長符号化スキームでの実際のクロック速度は「ボー」レートと称され、これはこのインタフェースのクロック速度とは無関係である。

【００１８】物理層５４はデータと制御ビットを転送するための双方向性のピンを制御する。リンク層５２は、制御が物理層５４からリンク層５２に移行されるとき、これらのピンを駆動するだけである。リンク層は線６４上の専用の要求ピンを介して全ての自発的な活動を行う。インタフェース上で起こり得るアクションは、送信、受信、状態および要求に分類される。システム・クロックＳＣＬＫは物理層５４によって駆動され、通常、４９．１５２ＭＨｚの速度でシリアル・バス・クロックと同期をとられる。「ハイ」に設定されると物理層がバックプレーン環境にあることを示すバックプレーン入力が、リンク層５２上に設けられている。他の入力としてＣＬＫ２５入力が設けられており、「ハイ」に設定されると線６６を介した物理層５４からのＳＣＬＫ出力を２４．５７６ＭＨｚの値にする。

【００１９】データが２つのチップ間を搬送されるとき、データ・バス６０の幅は接続された物理層５４の最大速度に依存し、１００メガビット／秒毎に２ビットである。そのため、毎秒１００メガビット転送のためのパケット・データはＤ[０：１]を用い、毎秒２００メガビット転送ではＤ[０：３]を用い、毎秒４００メガビット転送では最大限のＤ[０：７]を用いる。未使用のＤ[ｎ]信号は「ロー」に駆動される。制御バス６２は常に２ビットを搬送する。制御が物理層５４とリンク層５２の間で移行されるときはいつも、サイド・サレンダリング制御がなされる。それにより、制御ピンとデータ・ピンは、その出力バッファをトライステートする前に１クロックの間常に論理「０」レベルに駆動される。

【００２０】上述したように、要求、状態、送信および受信という４つの基本的な動作がインタフェースで起こり得る。バスを要求するには、または、物理層５４内のレジスタにアクセスするには、リンク層５２は短いシリアル・ストリームを要求ピン６４を介して物理層５４に送る。物理層５４がリンク層５２に転送すべき状態情報を有しているとき、物理層５４は状態転送を開始する。物理層５４はインタフェースがこの転送を行うのにアイドルになるまで待つ。そして、制御バス６２上に状態ビ

11

ットをアサートすることによって転送を開始する。同時に、状態情報の最初の２ビットをＤ[０：１]上に与える。リンク層５２が要求線６４を介してシリアル・バス５８へのアクセスを要求するとき、物理層５４はシリアル・バス５８へのアクセスについてアービトレーション（裁定）を行う。物理層５４は、裁定が成ると、システム・クロックＳＣＬＫの１サイクルの間制御バス６２上に「送信」をアサートすることによって、リンク層５２に対してシリアル・バスを許可する。その後、単一のサイクルの間アイドルである。リンク層５２は、物理層５４からの「送信」状態をサンプリングした後、制御バス６２上に「ホールド」あるいは「送信」をアサートすることによってインタフェースの制御を引き継ぐ。「受信」動作中は、物理層５４は、シリアル・バス５８に「データ・オン」状態を認めるときはいつも、制御バス６２上に「受信」をアサートしかつ各データ・ピン上に論理「１」をアサートすることによって受信動作を開始する。次いで、物理層５４はデータ・ピン上に速度符号を配することによってパケットのスタートを指示する。毎秒１００メガビットに対しては、データ・ビットは「００ｘｘｘｘｘｘ」である。また、毎秒２００メガビットに対しては「０１００ｘｘｘｘ」、毎秒４００メガビットに対しては「０１０１０００」である。ここで、値「ｘ」はノン・オペレーションである。

【００２１】リンク層５２はＦＩＦＯ７０の形のバッファにインタフェースしている。ＦＩＦＯ７０は、読み出し／書き込みポインタの位置とＦＩＦＯの全てのイン／アウト・アクセスを定める読み出し／書き込みＦＩＦＯ制御ブロック７１によって制御される。ＦＩＦＯ７０の他の側は、３２ビット・バスであるホスト・バス７２にインタフェースしている。

【００２２】次に図４を参照すると、パケットの非同期送信のためのリンク層５２でのサブアクションが示されている。このサブアクションは要求と応答の形である。パケットを送信したいノードによって送信される「ＡＲＢ」とラベルされたアービトレーションシーケンスがあり、このシーケンスが物理層５４に送信されてバス５８の制御を獲得する。物理層５４が既にバスを制御しているなら、物理層５４は直ちに応答でき、データ・パケットの送信が行なわれる。このデータ・パケットの送信は、非同期サブアクションでは、ソース・ノードがデータ・プリフィクス信号（必要なら速度符号を含む）、ソース・ノードおよび宛て先ノードのアドレス、トランザクション符号、トランザクション・ラベル、リトライ符号、データ、巡回冗長チェック（ＣＲＣ）およびパケット終了（他のデータ・プリフィクス信号かデータ終了信号）を送信することを含む。この後に、一意的にアドレスされた宛て先がパケットの送信先（受信側）でとられるアクションを送信元ノードに示す符号を返送する肯定応答（ＡＣＫ）フィールドが続く。これらの各非同期サブアクションは「サブアクション間隙」と呼ばれる空きバス期間で隔てられている。パケット送信と肯定応答受信の間に「肯定応答（ＡＣＫ）間隙」が設けられている。この肯定応答間隙の長さは受信側がリンク要求と肯定応答（ＡＣＫ）の送信元に相対してどこにあるかに依存して変わる。しかしながら、肯定応答間隙の最大長はサブアクション間隙よりは十分に短く、肯定応答が受信される前にバス上の他のノードがアービトレーションを開始しないよう保証する。

12

【００２３】次に図５を参照すると、リンク層５２が要求を扱う（要求に対してサービスを行う）様子の線図が示されている。上に述べたように、リンク層５２は要求、指示、応答および確認のサービス・プリミティブを利用する。要求プリミティブは要求側リンク層（リンク要求元）によって応答側リンク層（リンク応答側）へパケットを転送するために用いられる。指示プリミティブは応答側リンク層によるパケットの受信を示す。応答プリミティブは応答側リンク層による肯定応答の送信を示し、確認プリミティブは要求側リンク層による肯定応答の受信を示す。ひとたびリンク要求がなされると、システムは受信ノードへのアービトレーションおよびパケットの送信を行い、次いで、受信ノードが肯定応答の形で要求側リンク層に返答し、そこで要求側リンク層が送信を確認する。

【００２４】次に図６を参照すると、送信されるパケットのためのレジスタ・マップが示されている。パケットは複数のクワッドレットを含むヘッダを有する形態である。典型的には、最初のクワッドレットは物理ＩＤを含み、最後のクワッドレットはヘッダＣＲＣを含む。ヘッダ・パケットの後にはデータ・ブロックが続き、該データ・ブロックは複数のデータ・クワッドレットから成り、最後のクワッドレットはデータＣＲＣクワッドレットである。最初のコードレット内のパケット・ヘッダは、プライマリ・パケット（主パケット）のパケット種類を定めるトランザクション符号を含む。トランザクション・パケット符号はパケット・フォーマットと実行されるトランザクションの種類を特定する。これはデータ・クワッドレットの書き込み要求、データ・ブロックの書き込み要求、データ・クワッドレットおよびデータ・ブロックの読み出し要求、データ・クワッドレットおよびデータ・ブロックの読み出し応答等である。図４に関連して上で述べた非同期データ・パケットはプライマリ・データ・パケットである。

【００２５】次に図７ａを参照すると、本発明の好適実施例のブロック図が示されている。上で説明したように、物理層５４はリンク層５２とトランザクション層５０の両方にインタフェースしており、これは図２に示されている。本発明の好適実施例においては、リンク／トランザクション層コントローラ（Ｌ／Ｔコントローラ）２００が設けられており、該コントローラ２００は物理

層５４とアドレス２０４、データ・バス２０６および制御バス２０８から成る一組のホスト・システム・バス２０２との間でインタフェースするよう動作できる。ホスト・バス２０２は周辺装置２１０にインタフェースするよう動作できる。このコントローラ２００は、マイクロコントローラの動作を本質的にエミュレートすることに加えて、図２のリンク層５２、トランザクション層５０およびシリアル・バス・マネージャ５６の機能の全てを提供するよう構成されている。それゆえ、以下で説明するように、このコントローラ２００は、別個のマイクロプロセッサあるいはマイクロコントローラを必要とせずにシリアル・バス５８と周辺装置２１０との間に直接インタフェースできる。

【００２６】リンク層５２とトランザクション層５０の機能の全てに加えて、ホスト・バス２０２にインタフェースするためのインタフェースが設けられている。コントローラ２００はアドレス・ブロック２１４を用いてアドレス・バス２０４にアドレスを送信したり、アドレス・バス２０４からアドレスを受信するよう動作できる。データについてはデータ・バッファ２１６を介したデータ・バス２０６への送信およびデータ・バス２０６からの受信ができ、制御情報についてはコントローラ２１８を介した制御バス２０８への送信および制御バス２０８からの受信ができる。コントローラ２００内には内部レジスタ空間２２０があり、該内部レジスタ空間２２０はＩＥＥＥ１２１２ＣＳＲアドレス空間等の従来のアドレス空間中でコントローラ２００を定めるよう動作できる。コントローラ２００中の内部レジスタ空間２２０は他のノードのシステムによってアクセス可能で、直接これらのレジスタへの書き込みおよび読み出し要求を許す。また、内部ＦＩＦＯ２１９が設けられており、該内部ＦＩＦＯ２１９は、基本的には、データ・バッファ２１６と関連してシステムのホスト側とシリアル・バス側間のデータ転送のバッファを行うように働く。

【００２７】動作に際して、シリアル・バス上の他の場所にあるＣＰＵ（図示されていない）は、この特定のノードにどのような種類の装置があるかを決定するために内部レジスタ２２０にアクセスすることができる。ひとたびこのノードが周辺装置への直接アクセスを許す特別なアーキテクチャを有していることをＣＰＵが認識すると、本発明に従ってデータ転送が影響を受ける。もしコントローラ内の内部レジスタ中の情報が遠隔の場所で望まれているならば、送信装置からのデータ・パケットは、コントローラ２００で認識されることになるアドレス情報とデータ情報の両方からデータが構成されるように配置される。これは、アドレスとしてばかりでなく後で実行される書き込みまたは読み出し動作の要求として認識される。もし送信ノードが周辺装置２１０にアクセスすることを望むなら、受信アドレスのアドレス・バス２０４への転送と受信データのデータ・バス２０６を介した周辺装置２１０への転送を基本的に含む操作をコントローラ２００にさせるのに十分な情報をアドレスおよびデータ情報の形でコントローラ２００に送信するだけでよい。これは、周辺ユニットがコントローラ２００のアドレス空間の特定の部分を占有しており、そのようなアドレスが受け取られると、コントローラ２００は直ちにこのアドレスをホスト側用のアドレス・バス２０４に転送すること、および同時にデータをホスト・データ・バス上に置くことを、ホスト・システムのために適当な読み出しまたは書き込み指令を発生することに加えて行うという事実に因る。このことは、データが先ずＦＩＦＯに記憶されることを必要とし、次いで、トランザクション層５０によって関連したマイクロコントローラへの割り込みが発生する従来のトランザクションと比較されるべきである。その後、マイクロコントローラがこの割り込みに基づいて行動し、ＦＩＦＯから適当な情報を取り出したり引き続く処理を実行することになる。これに対して、本発明のシステムは、システム・バス２０２に直接インタフェースして、コントローラ２００が非同期データおよび等時性データの両方を自律的に受信し送信するのを可能にする。

【００２８】図７ｂに示されている本発明の別の実施例においては、コントローラ２００からのアドレス・バス２０４の一部とデータ・バス２０６が結合されている。好ましい実施例では、データ・バス２０６は８ビット・データ・バスで、アドレス・バス２０４は１２ビット・バスである。アドレス・バス２０４の８ビットとデータ・バス２０６の全８ビットを利用することによって、１６ビット・データ・バスとしての新たなデータ・バス２１１が実現できる。このバスが１６ビットの周辺装置２０９に入力し、両者間で１６ビット・データの転送が可能になる。これは、或る応用において１６ビット・データの転送を可能にする独特なモードである。これは単に、限られた数のピンがチップ上に設けられているという事実に因る。また、遠隔ノードがアドレスおよびデータ・バス上に出力されるべき情報を送っているのをコントローラ２００が認識するという事実によって、データはアドレス・バス２０４の最下位７ビットとデータ・バス２０６の全データ・ビットを介して１６ビット・データ・バス２１１に転送されるのがわかる。コントローラ２００は実際にはアドレス指定動作をしていないし、アドレス・バス２０４やデータ・バス２０６上に何があるかについて何の判定もしていないということ、それよりむしろ、遠隔ノードがコントローラ２００に特別な様式でデータおよびアドレス情報を送っている結果としてコントローラ２００はこのデータおよびアドレス情報をそれら２つのバスに単に転送しているだけであるということに注目することが重要である。遠隔ノードがアドレス・バス２０４およびデータ・バス２０６への情報をどのように得るかを認識することが単に重要である。以下で

より詳細に説明するように、このことは、特定の内部レジスタに情報を送信することだけを遠隔ノードに要求するだけであり、結果として、コントローラ２００が自動的にこの情報を適当なバス上に出力する。遠隔ノードは該ノードが情報を送信している先の装置の種類についての知識を持っているので、コントローラ２００に送信された情報が自動的にアドレス・バス２０４およびデータ・バス２０６に直接転送されるという事実が分かる。

【００２９】再び図７ａを参照すると、データはホスト・システム・バス２０２とシリアル・バス５８との間を２つの方法のうちの何れかで転送される。第１の方法においては、データは内部レジスタに転送され、直ちにアドレス送受信ブロック２１４およびデータ・バッファ２１６に転送され、そして直ちにデータ・バス２０６およびアドレス・バス２０４上に置かれる。このように、遠隔ノードは情報を直接システム・バス２０２に転送することができる。他のモード（第２の方法）においては、データは内部ＦＩＦＯ２１９にそこへの記憶のため転送され、そして後で周辺装置２１０に転送される。送信動作では、データはホストまたはシステム・バス２０２から受信され、ＦＩＦＯ２１９にストア・アンド・フォワード（記憶／送り出し）様式で記憶される。次いでシリアル・バス上に送信される（送り出される）。

【００３０】次に図８を参照すると、コントローラ２００のディジタル・カメラ内での応用が示されている。ディジタル・カメラはイメージャ２５０と該イメージャ２５０にインタフェースしてそこからデータを受信するフィールド・メモリ２５２とを備えている。カメラはまた、それに関連付けられた制御論理ブロック２５４を有しており、該制御論理ブロック２５４はそこにデータが送られてくるとコントローラ２００に対して線２５６上に送信事象信号ＴＥＶＥＮＴを発生するよう動作可能である。コントローラ２００は、この信号を受け取ると、フィールド・メモリ２５２内のデータにアクセスしてこの情報を直接シリアル・バス５８そして遠隔の場所に送信する。上で言及したように、この動作がどのように実行されまたどのノードに送信されるかを決定するようプログラムできる内部レジスタ２２０が設けられている。

【００３１】次に図９を参照すると、本発明の他の応用としてオーディオ送信器への応用が示されている。アナログ入力信号はＡ／Ｄ（アナログ／ディジタル）変換器２６２に入力する線２６０上のステレオ入力信号である。Ａ／Ｄ変換器２６２はテキサス・インスツルメンツ製のタイプＴＬＣ３２０ＡＤ５７Ｃである。Ａ／Ｄ変換器２６２はコントローラ２００のデータ入力への線２６４上にステレオ・ビット・シリアル・ストリームを発生する。制御論理ブロック２７０は情報がコントローラ２００に送信されているのを示す送信事象信号ＴＥＶＥＮＴを線２７２上に発生するのに用いられる。コントローラ２００は制御論理ブロック２７０にインタフェースしていて、データの発生とＦＩＦＯ２１９内への記憶のためのコントローラ２００への送信を制御し、その後このデータをシリアル・バス５８に送信する。

【００３２】次に図１０を参照すると、コントローラ２００の他の応用が示されている。ここでは、コントローラ２００はオーディオ受信器として利用されている。コントローラ２００はシリアル・バス５８からデータを連続的に受信する。このデータは連続して複号されてＦＩＦＯ２１９に記憶され、さらにデータ／アドレス線２７６上をＤ／Ａ（ディジタル／アナログ）変換器２７８に送られ、そこからアナログ・ステレオ出力信号として線２７９上に出力される。制御論理ブロック２８２がコントローラ２００にインタフェースするように設けられている。データ／アドレス情報はデータ８ビット、アドレス情報８ビットで伝送される。

【００３３】次に図１１を参照すると、遠隔送信ノード２８０がローカル・ノード２８２に情報を送信するよう動作できる一般的な応用のブロック図が示されている。ローカル・ノード２８２は、上で説明したコントローラ２００と遠隔ノード２８０からの情報が送られるべき周辺ユニット２８４とを備えて構成されている。遠隔ノード２８０は物理層２８８に加えてリンク層２８６とトランザクション層（図示されていない）をそれに関連付けられて有している。そのため、物理層２８８とリンク層２８６は、遠隔ノード２８０がシリアル・バス５８を越えてローカル・ノード２８２と通信を行って、そこにデータ・パケットを送ったり、そこからデータ・パケットを受け取ったりするのを可能にする。遠隔ノード２８０にはまた、ローカル・ノード２８２にある周辺ユニット２８４と通信するためＣＰＵ２９０が設けられている。上で説明したように、ＣＰＵ２９０はバス５８上の複数のノードと通信可能である。ノード２８２と通信するには、バス５８上にどんな種類のノードが存在しているかを先ず知らなければならない。このために、ＣＰＵ２９０はコントローラ２００のアドレス空間（これは典型的なＣＳＲ１２１２アドレス空間である）内の既知のレジスタ位置にアクセスする。ローカル・ノード２８２はこのアドレス空間を有して構成されているが、他のノードも全てＣＳＲ１２１２アドレス空間を有して構成されている。コントローラ２００内の特定のアドレス位置にアクセスするとき、ＣＰＵ２９０はローカル・ノード２８２上の他のＣＰＵに行く必要はない。というよりはむしろ、そのような要求がなされるとＣＰＵ２９０はコントローラ２００の内部レジスタに直接行く。そうすると、コントローラ２００は要求された情報をシリアル・バス５８に送信する。ＣＰＵ２９０はコントローラ２００がローカル・ノード２８２に存在することを認識すると、ＣＰＵ２９０は通常とは若干異なる様式で読み出しおよび書き込み要求を発生する。ＣＰＵ２９０は、ＦＩＦＯを通る必要なく周辺ユニット２８４のアドレス空間への

直接アクセスが仮想的にできることを知って、それらの要求を送信する。これについては後で説明する。このことは、実際においては、通常はローカル・ノードにあるＣＰＵの機能性の幾らかを遠隔ノードに移す。さらに、遠隔ノードのＣＰＵ２９０はコンフィギュレーション（環境設定）命令をコントローラ２００の内部レジスタに直接送信してこれらのレジスタをコンフィギュアすることができる。

【００３４】遠隔ノードのＣＰＵ２９０にローカル・ノードのコントローラ２００をプログラミングするのを許すために、複数のコンフィギュレーション（環境設定）レジスタがコントローラ２００のアドレス空間にマッピングされる。これらのコンフィギュレーション・レジスタはホスト・インタフェースから直接アクセス可能である。一般に、ＩＥＥＥ１３９４バス・プロトコルは６４ビット固定アドレッシング・スキームを用いている。この６４ビット・アドレスは１０ビットのバス番号と、６ビットのノード番号と、２０ビットのページ・アドレスと、２８ビットのオフセットから成る。各ノードについて３つのアドレス可能なメモリ空間がある。それらはメモリ空間、プライベート空間およびレジスタ空間である。各空間のアドレスを表１に示す。

【００３５】

【表１】

表　1

メモリ	Bus_Node_00000_0000000 Bus_Node_FFFFD_FFFFFFF
ﾌﾟﾗｲﾍﾞｰﾄ	Bus_Node_FFFFE_0000000 Bus_Node_FFFFE_FFFFFFF
レジスタ	Bus_Node_FFFFF_0000000 Bus_Node_FFFFF_FFFFFFF

【００３６】６４ビット・アドレッシング・スキーム用のアドレスは図１２の線図に従って構成される。物理層がシリアル・バス５８から情報を抽出するには物理層に十分な情報を送る必要があることが図１２から分かる。一度ストリップされると、この情報の全てをリンク／トランザクション層コントローラ２００に送信する必要はない。ページ・アドレスと２８ビットのオフセット情報（この２８ビットのオフセット情報は内部レジスタ空間を定める）を送信するだけでよい。

【００３７】次に図１３を参照すると、所与のコントローラ２００の内部レジスタ空間についてのアドレス空間の線図が示されている。コントローラ２００をＩＥＥＥ１２１２ＣＳＲアドレス空間に合わせるには、或る情報がアドレス空間内の特定の位置にあることを保証する必要がある。それゆえ、ＣＳＲアドレス空間は、位置０－５１２にＣＳＲアーキテクチャに関する情報、位置５１２－１０２４にシリアル・バスに関する情報、位置１０２４－２０４８にＲＯＭ（リード・オンリ・メモリ）位置（これらは必ずしも変更されるとは限らない）、および位置２０４８－４０９６に特定のコントローラ・ユニット自体に関する情報を有し、残りの位置４０９６－２５６Ｍは未使用ユニット空間として残してある。

【００３８】所与の領域内のメモリ位置の全てが機能部分を提供する必要はない。そのため、必要最小量の情報だけが記憶される。例えば、ＣＳＲアドレス空間アーキテクチャについては、ＳＴＡＴＥ＿ＣＬＥＡＲ（状態クリア）、ＳＴＡＴＥ＿ＳＥＴ（状態設定）、ＮＯＤＥ＿ＩＤＳ（ノード識別子）およびＲＥＳＥＴ＿ＳＴＡＲＴ（リセット・スタート）の４つのレジスタだけが設けられている。シリアス・バス領域については、ＣＹＣＬＥ＿ＴＩＭＥ（サイクル・タイム）およびＢＵＳＹ＿ＴＩＭＯＵＴ（ビジー・タイムアウト）に関する情報だけが２つのレジスタに設けられている。リード・オンリ・メモリ（ＲＯＭ）情報とみなされる情報については、ＭＩＮＩＭＡＬ＿ＲＯＭ（最小ＲＯＭ）と称する位置に記憶される。これは、一般には、製造者、部品および部品に関連したタイプ（種類）に関する情報である。ユニット空間に関連した位置には、ＶＥＲＳＩＯＮ（バージョン）、ＩＮＴＥＲＲＵＰＴ（割り込みの種類）、ＩＮＴＥＲＲＵＰＴ＿ＭＡＳＫ（割り込みマスクの種類）、およびＰＨＹ＿ＣＨＩＰ＿ＡＣＣＥＳＳ（物理層に対する種々のチップ・アクセス情報）等が記憶される。それに加えて、Ｔ＿ＣＯＮＴＲＯＬ＿ＲＥＧ０－４と標示した複数の送信制御レジスタが設けられている。これらのレジスタについては、後でより詳細に説明する。

【００３９】コントローラ２００は、シリアル・バス５８から要求を受信するときはいつも全６４ビットアドレスを複号する。もし複号されたアドレスがＣＳＲレジスタ空間内のアドレスとオン・チップ・レジスタ（即ち、図１３で記憶情報を持つものとして説明したレジスタ）への下位１２ビット点を構成するなら、そのレジスタがシリアル・バス５８からアクセスされる。ＣＳＲ１２１２レジスタ空間のために取って置かれた占有されていない多数の位置があることが注意されるべきである。この理由は最小組のレジスタだけが設けられているということである。もし、何らかの理由で、アドレス空間のアンポピュレイテッド（非居住または非分布）部分に通常見いだされる命令あるいは情報、即ち、コントローラ２００に含まれていない命令あるいは情報が遠隔ノードから動作するよう求められたならば、そのアドレス空間を占めるローカル・ノードの外部レジスタが必要とされる。書き込み動作がローカル・ノードのコントローラ２００にあるオン・チップ・レジスタに向けられているときはいつも、これらの書き込みは、もし内部エコー・ビットＣＳＲ＿ＷＲ＿ＥＣＨＯが設定されているなら、ホスト・インタフェースに反映される。ホスト・インタフェース・アドレス（Ａ［１１：０］）は単純に６４ビット・シリアル・バス・アドレスの１２ＬＳＢ（最下位１２ビット）である。オン・チップ・レジスタにアクセスする特別な場合は、後で説明するＭｉｃｒｏ＿Ｍ＿Ｒｅｇにアクセスする場合である。もしレジスタ空間がオン・チ

ップでないレジスタへの下位１２ビット点に複号されているならば、コントローラ２００はこれを認識する。そして、ＣＳＲ１２１２がアプリケーションによって履行されていると仮定して、読み出しと書き込みの両方がホスト・インタフェース上で実行される。ＣＳＲ１２１２レジスタ空間は単に１つの標準であり、遠隔ノードが何れかのノードにあるコントローラ２００で利用されている特定の標準を知っている限り、任意のレジスタ空間が利用できることが理解されるべきである。本アプリケーションでは、ＣＳＲ１２１２は部品を周知の標準に合わせるのに用いられている。

【００４０】一般に、チップに収容されたホストの種類は、ホストのタイプを定めるＨＴＹＰと標示された外部ハードウエア・ピンによって定義される。もしホストの種類が８ビット・マスタに設定されるなら、システムはマイクロコントローラをエミュレートでき、ホスト・システムをインタフェースできる。もしホストの種類が単にＣＳＲアクセス・モードに設定されるなら、位置０－４０９６間の内部ＣＳＲだけがアクセス可能である。

【００４１】コントローラ２００について、次の４つの基本的な動作モード、即ち、

１．非同期受信、

２．非同期送信、

３．等時性送信、および

４．ホストＣＳＲアクセスがある。

【００４２】サポートされるトランザクションの種類およびホスト・インタフェース・トランザクションへのマッピングの概要を表２に示す。

【００４３】

【表２】

【表3】

表 2

HTYPE	受信 1394 トランザクション	ホスト・インタフェース・トランザクション	ack	送信 1394トランザクション	AADEC
0	Micro_M_Reg へのコードレット書き込み要求 dest_addr = Node_id, FFFFF,Micro_M_Reg	8ビット・マスタ・バイト書き込み A[11:0] = quadlet_data[12:23] D[7:0] = quadlet_data[24:31]	1 2 1	Write_Resp_En が設定されていなければ、書き込み要求なし。 Write_Resp_En が設定なら、CAZ 上に書き込み要求。 Write_Resp_Cat が設定なら、書き込み要求が連結。	0
1,2	Micro_M_Reg へのコードレット書き込み要求 dest_addr = Node_id, FFFFF,Micro_M_Reg	無	4	無	0
0	Micro_M_Reg へのコードレット読み出し要求 dest_addr = Node_id, FFFFF,Micro_M_Reg	8ビット・マスタ・バイト 読み出し A[11:0] = quadlet_data[12:23] response quadlet_data[24:31] = D[7:0]	2 1	CAZ 上に 読み出し要求。 Read_Resp_Cat が設定なら、読み出し要求が連結。	0
1,2	Micro_M_Reg へのコードレット読み出し要求 dest_addr = Node_id, FFFFF,Micro_M_Reg	無	4	無	0
0	Initial Memory Space へのコードレット書き込み要求 Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	4回の8ビット・マスタ・バイト書き込み (順番は 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31]	1 2 1	Write_Resp_En が設定されていなければ、書き込み要求なし。 Write_Resp_En が設定なら、CAZ 上に書き込み要求。 Write_Resp_Cat が設定なら、書き込み要求が連結。	1
1	Initial Memory Space へのコードレット書き込み要求 Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	16ビット・スレーブ・モード 1回目のアクセス A[7:0] = quadlet_data[0:7] D[7:0] = quadlet_data[8:15] 2回目のアクセス A[7:0] = quadlet_data[16:23] D[7:0] = quadlet_data[24:31]	1 2 1	Write_Resp_En が設定されてなければ、書き込み要求なし。 Write_Resp_En が設定なら、CAZ 上に書き込み要求。 Write_Resp_Cat が設定なら、書き込み要求が連結。	1

表 2 の 続 き

HTYPE	受信 1394 トランザクション	ホスト・インタフェース・トランザクション	ack	送信 1394トランザクション	AADEC
1	Initial Memory Space へのブロック書き込み要求 Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	16ビット・スレーブ・モード、CAZが最終データを示すまでホスト・インタフェースからデータ読み出し。 1回目アクセス A[7:0] = quadlet_data[0:7] D[7:0] = quadlet_data[8:15] 2回目アクセス A[7:0] = quadlet_data[16:23] D[7:0] = quadlet_data[24:31] 引き続くアクセス A[7:0] = next quadlet_data[0:7] D[7:0] = next quadlet_data[8:15]	1 2 1	Write_Resp_En が設定されていなければ、書き込み要求なし。 Write_Resp_En が設定なら、CAZ 上に書き込み要求。 Write_Resp_Cat が設定なら、書き込み要求が連結。	1
2	Initial Memory Space へのコードレットまたはブロック書き込み要求 Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	無	4	無	1
0	Initial Memory Space へのコードレット読み出し要求 Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	4回の 8ビット・マスク・バイト 読み出し (順番は 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31]	2 1	CAZ 上に 読み出し要求。 Read_Resp_Cat が設定なら、読み出し要求が連結。	1
1	Initial Memory Space へのコードレット読み出し要求 Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	16ビット・スレーブ・モード 1回目のアクセス A[7:0] = quadlet_data[0:7] D[7:0] = quadlet_data[8:15] 2回目のアクセス A[7:0] = quadlet_data[16:23] D[7:0] = quadlet_data[24:31]	2 1	CAZ 上に 読み出し要求。 Read_Resp_Cat が設定なら、読み出し要求が連結。	1
1	Initial Memory Space へのブロック読み出し要求 Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	16ビット・スレーブ・モード、CAZが最終データを示すまでホスト・インタフェースからデータ読み出し。 1回目アクセス A[7:0] = quadlet_data[0:7] D[7:0] = quadlet_data[8:15] 2回目アクセス A[7:0] = quadlet_data[16:23] D[7:0] = quadlet_data[24:31] 引き続くアクセス A[7:0] = next quadlet_data[0:7]	1 2 1	Write_Resp_En が設定されていなければ、書き込み要求なし。 Write_Resp_En が設定なら、CAZ 上に書き込み要求。 Write_Resp_Cat が設定なら、書き込み要求が連結。	1

【表４】

表２の続き

		D[7:0] = next quadlet data[8:15]			
2	Initial Memory Space へのｺｰﾄﾞﾚｯﾄまたはﾌﾞﾛｯｸ読み出し要求 Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFF, 0000000	無	4	無	1
0	CSR へのｺｰﾄﾞﾚｯﾄ書き込み要求 (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, は FFFFF,FFFFFFF) dest_addr 内部ﾚｼﾞｽﾀ組中になし。	4回の8ﾋﾞｯﾄ･ﾏｽｸ･ﾊﾞｲﾄ書き込み (順番は 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31]	1 2 1	Write_Resp_En が設定されていなければ、書き込み要求なし、 Write_Resp_En が設定なら、CAZ 上に書き込み要求、 Write_Resp_Cat が設定なら、書き込み要求が連結。	2
1,2	CSR へのｺｰﾄﾞﾚｯﾄ書き込み要求 (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, は FFFFF,FFFFFFF) dest_addr 内部ﾚｼﾞｽﾀ組中になし。	無	4	無	2
0	CSR へのｺｰﾄﾞﾚｯﾄ読み出し要求 (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, は FFFFF,FFFFFFF) dest_addr 内部ﾚｼﾞｽﾀ組中になし。	4回の8ﾋﾞｯﾄ･ﾏｽｸ･ﾊﾞｲﾄ読み出し (順番は 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31]	2 1	CAZ 上に読み出し要求。 Read_Resp_Cat が設定なら、読み出し要求が連結。	2
1,2	CSR へのｺｰﾄﾞﾚｯﾄ読み出し要求 (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, は FFFFF,FFFFFFF) dest_addr 内部ﾚｼﾞｽﾀ組中になし。	無	4	無	2
0	CSR へのｺｰﾄﾞﾚｯﾄ書き込み要求 (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) dest_addr は 内部ﾚｼﾞｽﾀ組中にあり。	CSR_WR_ECHO ﾋﾞｯﾄが設定なら 4回の8ﾋﾞｯﾄ･ﾏｽｸ･ﾊﾞｲﾄ書き込み(順番は 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31] CSR_WR_ECHO ﾋﾞｯﾄが設定なら、ｽﾃｰﾀｽ･ﾄﾗﾝｻﾞｸｼｮﾝなし。	1 2 1	Write_Resp_En が設定されていなければ、書き込み要求なし、 Write_Resp_En が設定なら、CAZ 上に書き込み要求、 Write_Resp_Cat が設定なら、書き込み要求が連結。	3
1,2	CSR へのｺｰﾄﾞﾚｯﾄ書き込み要求 (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) dest_addr は 内部ﾚｼﾞｽﾀ組中にあり。	無	4	無	3
0	CSR へのｺｰﾄﾞﾚｯﾄ読み出し要求 (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) dest_addr は 内部ﾚｼﾞｽﾀ組中にあり。	無	2 1	CAZ 上に読み出し要求。 Read_Resp_Cat が設定なら、読み出し要求が連結。	3
1,2	CSR へのｺｰﾄﾞﾚｯﾄ読み出し要求 (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) dest_addr は 内部ﾚｼﾞｽﾀ組中にあり。	無	4	無	3

【００４４】上記４つの基本的動作モードの各々について以下に詳細に説明する。

【００４５】非同期受信

マイクロコントローラ・エミュレーション・モード、ＡＡＤＥＣ[１：０]＝０では、コントローラ２００はＣＳＲ Ｍｉｃｒｏ＿Ｍ＿Ｒｅｇへのシリアル・バス書き込みまたは読み出し要求トランザクションに応答してホスト・インタフェース上に８ビット書き込みおよび読み出しトランザクションを発する。Ｍｉｃｒｏ＿Ｍ＿Ｒｅｇへの書き込み要求は、コントローラ２００にそのホスト・インタフェース上でアドレスＡＭ、データＤＭでの書き込みトランザクションを実行させる。Ｍｉｃｒｏ＿Ｍ＿Ｒｅｇに書き込まれたデータの最下位バイトはＤ[７：０]上にアサートされ、最下位バイトの次はＡ[１１：０]上にアサートされる。ＣＳＺ、ＣＡＺ、ＷＥＺ、ＡＡＤＥＣおよびＨＴＹＰＥは図１７に示されたタイミングでのトランザクションの要素である。Ｍｉｃｒｏ＿Ｍ＿Ｒｅｇへの読み出し要求は、コントローラ２００にそのホスト・インタフェース上で読み出しトランザクションを実行させる。読み出しがＣＡＺで肯定応答されると、コントローラ２００は読み出し応答を送信する。もしＣＡＺがＣＳＺの０．９マイクロ秒以内で生ずると、読み出し応答は対応する要求肯定応答に連結される。１つの顕著な読み出し要求だけが許可される。この動作のタイミングが図１８に示されている。

【００４６】メモリ・モード、ＡＡＤＥＣ[１：０]＝１では、コントローラ２００は８コードレット内部ＦＩＦＯを用いてノードの初期メモリ空間にアドレスされたシリアル・バス・コードレットまたはブロック読み出しまたは書き込み要求を扱う。ホスト・インタフェース／Ｆ

ＩＦＯ転送は８ビット・マスタか１６ビット・スレーブの何れかである。

【００４７】８ビット・マスタ書き込み要求の間、初期メモリ空間への書き込み要求（コードレットまたはブロック）はＦＩＦＯにデータを記憶する。コントローラ２００は、図１９に示されたＳＲＡＭ様のタイミングでＡ、Ｄ、ＷＥＺ、ＣＳＺおよびＣＡＺを用いたバイト・サイズの書き込みトランザクションを行う。Ａ［１１：０］上にアサートされるアドレスは書き込み要求の目的（宛て先）アドレスの最下位１２ビット（１２ＬＳＢ）である。

【００４８】８ビット・マスタ読み出し要求の間、コントローラ２００は初期メモリ空間への読み出し要求を受け取り、Ａ、Ｄ、ＷＥＺ、ＣＳＺおよびＣＡＺを用いたバイト・サイズのトランザクション（ＦＩＦＯへのデータ読み出し）を行う。ＦＩＦＯへの読み出しが完了すると、コントローラ２００は読み出し応答を送信する。この動作のタイミングが図２０に示されている。Ａ［１１：０］上にアサートされるアドレスは読み出し要求の目的（宛て先）アドレスの最下位１２ビットである。

【００４９】１６ビット・スレーブ書き込み要求の間、書き込み要求（コードレットまたはブロック）はＦＩＦＯ２１９にデータを記憶する。その後、データは１６ビット・データ・バスとしての｛Ａ［７：０］、Ｄ［７：０］｝を用いて外部エージェントによってＦＩＦＯ２１９から読み出される。ＲＥＺは、書き込み要求が受信されかつそのデータがＦＩＦＯ２１９で入手可能であることを示す。ＣＳＺは、外部マスタがデータを受信する準備ができていることを示すために用いられている。データがコントローラ２００からクロック・アウトされる毎に、内部ダブレット（二重）ＦＩＦＯポインタがインクリメントされる。内部ダブレット・ポインタが要求されたデータ長に等しくなると、ＣＡＺが最終データを示すために用いられ、そして書き込み応答が送信される。この動作のタイミングが図２１に示されている。

【００５０】１６ビット・スレーブ読み出し要求では、コントローラ２００が初期メモリ空間への読み出し要求を受け取ると、応答データが１６ビット・データ・バスとしての｛Ａ［７：０］、Ｄ［７：０］｝を用いてＦＩＦＯ２１９内に書き込まれなければならない。ＷＥＺは、読み出し要求が受信されかつその空間がＦＩＦＯで入手可能であることを示す。ＣＳＺは、外部マスタがデータを発していることを示すのに用いられている。データがコントローラ２００にクロック・インされる毎に、内部ダブレットＦＩＦＯポインタがインクリメントされる。内部ダブレット・ポインタが要求されたデータ長に等しくなると、ＣＡＺが最終データを示すために用いられ、そして読み出し応答が送信される。この動作のタイミングが図２２に示されている。

【００５１】外部ＣＳＲモード、ＡＡＤＥＣ［１：０］＝２では、コントローラ２００中に含まれていない初期レジスタ空間および初期ユニット空間内のレジスタへの書き込み要求および読み出し要求が、８ビット・マスタ・メモリ・モードと同じバイト・サイズのＳＲＡＭ様トランザクションを用いて扱われる。

【００５２】内部ＣＳＲモード、ＡＡＤＥＣ［１：０］＝３では、コントローラ２００中に含まれている初期レジスタ空間および初期ユニット空間内のレジスタへの書き込み要求および読み出し要求が、ＣＳＲ＿ＷＲ＿ＥＣＨＯビットが設定されていなければ、ホスト・インタフェース上での活動なしに内部で実行される。この場合、コードレット書き込み要求の結果、コードレットは、８ビット・マスタ・メモリ・モードと同じバイト・サイズのＳＲＡＭ様トランザクションを用いて、ホスト・インタフェース上に反映されることになる。

【００５３】非同期送信

コントローラ２００は、非同期書き込み要求を自動的に規則的間隔であるいは手動で送信するようにプログラムできる。データ・コードレットおよびデータ・ブロック・フォーマットに対する書き込み要求だけがサポートされる。自動（またはセンサ）モードでは、８ビット・マスタ・モードのホスト・インタフェースが１から８コードレットのバケット・データをＦＩＦＯ２１９内に読み出す。このモードはストア・アンド・フォワード（記憶／送り出し）であり、データ読み出し量はＴ＿Ｃｏｎｔｒｏｌ＿Ｒｅｇ３内のＤａｔａ＿Ｌｅｎｇｔｈ（データ長）フィールドによって決定される。この読み出しはＳｅｎｓｏｒ＿Ｐｒｅｆｅｔｃｈ＿Ｅｖｅｎｔ（センサ先取り事象）で始まる。次に、コントローラ２００はＴ＿Ｃｏｎｔｒｏｌ＿Ｒｅｇ（０：３）内の情報を用いて非同期ヘッダをバケットにプリペンド（先掛け）し、組み立てた非同期書き込み要求をＳｅｎｓｏｒＴｒａｎｓｍｉｔ＿Ｅｖｅｎｔ（センサ送信事象）で送信する。ＳｅｎｓｏｒＰｒｅｆｅｔｃｈ＿Ｅｖｅｎｔは、Ｔ＿Ｃｏｎｔｒｏｌ＿Ｒｅｇ０のＳｅｎｓｏｒ＿Ｐｒｅｆｅｔｃｈ＿Ｅｖｅｎｔ＿Ｓｅｌｅｃｔ（ＳＰＥＳ：センサ先取り事象選択）フィールドを用いて内部事象あるいは外部事象の何れかで起こるようにプログラムできる。

【００５４】Ｓｅｎｓｏｒ＿Ｐｒｅｆｅｔｃｈ＿Ｅｖｅｎｔ＿Ｓｅｌｅｃｔ（センサ先取り事象選択）オプション：

１．センサ先取りディスエーブルド、
２．内部サイクル・スタート、
３．内部サイクル・ダン、および
４．ＴＥＶＥＮＴピンの外部立ち上がりエッジ。

【００５５】Ｓｅｎｓｏｒ＿Ｔｒａｎｓｍｉｔ＿Ｅｖｅｎｔ（センサ送信事象）も、Ｔ＿Ｃｏｎｔｒｏｌ＿Ｒｅｇ０のＳｅｎｓｏｒ＿Ｔｒａｎｓｍｉｔ＿Ｅｖｅｎｔ＿Ｓｅｌｅｃｔ（ＳＴＥＳ：センサ送信事象選択）フィールドを用いて類似の事象で起こるようにプログラムでき

る。

【００５６】Ｓｅｎｓｏｒ＿Ｔｒａｎｓｍｉｔ＿Ｅｖｅｎｔ＿Ｓｅｌｅｃｔ（センサ送信事象選択）オプション：

１．センサ送信ディスエーブルド、

２．内部サイクル・スタート、

３．内部サイクル・ダン、および

４．ＴＥＶＥＮＴピンの外部立ち下がりエッジ。

【００５７】手動非同期送信モードでは、１６ビット・スレーブ・モードのホスト・インタフェースがコードレットまたはブロック・データをＦＩＦＯ２１９中にロードするために用いられる。このモードはフロー・スルーであり、送信パイプライニングが生じ得る。ＦＩＦＯにロードされるデータ量がＴｘ＿Ｔｈｒｅｓｈｏｌｄ（Ｔｘしきい値）に達するかあるいはＭａｎｕａｌ＿Ｔｒａｎｓｍｉｔ＿Ｅｖｅｎｔ（手動送信事象）が生じると、コントローラ２００はＴ＿Ｃｏｎｔｒｏｌ＿Ｒｅｇ（０：３）内の情報を用いて非同期ヘッダをパケットにプリペンド（先掛け）し、書き込み要求の送信を始める。外部ホスト・インタフェース・マスタは送信速度と同速度を保たなければならない、あるいはＦＩＦＯアン＊ダーランが起こる。ＦＩＦＯオーバーランを管理するためにフロー制御が設けられる。これは図２１のタイミング図に示されている。

【００５８】Ｍａｎｕａｌ＿Ｔｒａｎｓｍｉｔ＿Ｅｖｅｎｔ（ＭＴＥ：手動送信事象）は、Ｔ＿Ｃｏｎｔｒｏｌ＿Ｒｅｇ０のＭａｎｕａｌ＿Ｔｒａｎｓｍｉｔ＿ＥｖｅｎｔＳｅｌｅｃｔ（ＭＴＥＳ：手動送信事象選択）フィールドを用いて起こるようにプログラムできる。

【００５９】Ｍａｎｕａｌ＿Ｔｒａｎｓｍｉｔ＿Ｅｖｅｎｔ＿Ｓｅｌｅｃｔ（手動送信事象選択）オプション：

１．内部サイクル・スタート、

２．内部サイクル・ダン、

３．Ｍａｎｕａｌ＿Ｔｒａｎｓｍｉｔ＿Ｅｖｅｎｔ（手動送信事象）ビットを「１」に設定、および

４．ＴＥＶＥＮＴピンの外部立ち下がりエッジ。

【００６０】４つの送信制御レジスタは図１４に示されたような複数のフィールドを有している。これらのフィールドの定義を表３に示す。

【００６１】

【表５】

表　3

フィールド・ニーモニック	フィールド幅	フィールド記述
STES	2	センサ送信事象選択
SPES	2	センサ先取り事象選択
MTES	2	手動送信事象選択
MTE	1	手動送信事象
tl	6	トランザクション・ラベル
rt	2	リトライ符号
tcode	4	トランザクション符号
pri	4	優先順位
destination_ID	16	アドレス
destination_offset_lo	16	アドレス
destination_offset_hi	32	アドレス
data_length	16	非同期および等時性送信用に用いられるデータ長
DMEN	2	データ・ムーバ・イネーブル 00＝ 自動非同期モード 01＝ 手動非同期モード 10＝ 自動等時性モード 11＝ 手動等時性モード
Tx_Threshold	2	送信しきい値 00＝2 コードレット(8バイト) 01＝4コードレット(16バイト) 10＝6コードレット(24バイト) 11＝8コードレット(32バイト)
tag	2	同時性ヘッダ・タグ
channel	6	同時性ヘッダ・チャネル番号
sy	4	同時性ヘッダ同期ビット

【００６２】等時性送信

コントローラ２００は、上で説明した非同期送信とほぼ同じ方法で等時性データ・ブロック・パケットを送信するようにプログラムできる。Ｔ＿Ｃｏｎｔｒｏｌ＿Ｒｅｇ４のＤａｔａ＿Ｍｏｖｅｒ＿Ｅｎａｂｌｅ（ＤＭＥＮ：データ・ムーバ・イネーブル）フィールドが自動または手動等時性モードに設定されると、コントローラ２００は、等時性ヘッダがＴ＿Ｃｏｎｔｒｏｌ＿Ｒｅｇ３およびＴＣｏｎｔｒｏｌ＿Ｒｅｇ４内の情報を用いてプリペンド（先掛け）される点を除いて、非同期送信モードと同様に機能する。

【００６３】ホストＣＳＲアクセス

31

ＨＴＹＰＥ［１：０］上に｛１０｝をアサートすると、コントローラ２００のホスト・インタフェースがＣＳＲアクセス・モードに置かれる。このモードでは、ホスト・インタフェースはスレーブとして働く。内部ＣＳＲレジスタに書き込みや読み出しができる。アドレスＡ［１１：０］が４Ｋの内部ＣＳＲ空間をアドレスするために用いられる。サイクル・スタート（ＣＡＺ）がトランザクションをスタートさせる入力として用いられる。書き込み（ＷＥＺ）が書き込みまたは読み出しサイクル肯定応答を示す入力として用いられる。（ＣＡＺ）がコントローラ２００によってアサートされ、トランザクションを終える。この動作のタイミングが図１５に示されている。

32

＊

【００６４】次に図１６を参照すると、コントローラ２００の線図が示されており、そこには物理リンク層５４との種々のインタフェースおよびホスト・インタフェース側の種々のアドレス、データおよび制御ピンが示されている。同図に見られるように、バス２０４用に１２ビット・アドレス・バスそしてバス２０６用に８ビット・データ・バスが設けられている。種々の端子機能を表４に示す。

【００６５】

【表６】

表　4

信号名称	ピン #	I/O	記述
A[11:0]		I/O	8ﾋﾞｯﾄ・ﾏｽﾀおよびCSRｱｸｾｽ・ﾎｽﾄ・ｲﾝﾀｰﾌｪｰｽ・ﾓｰﾄﾞでのｱﾄﾞﾚｽ。
D[7:0]		I/O	全てのﾎｽﾄ・ｲﾝﾀｰﾌｪｰｽ・ﾓｰﾄﾞでのﾃﾞｰﾀ。
CSZ		I/O	8ﾋﾞｯﾄ・ﾏｽﾀ・ﾓｰﾄﾞでの出力ｻｲｸﾙ・ｽﾀｰﾄ/ﾁｯﾌﾟ選択。 CSRｱｸｾｽ・ﾓｰﾄﾞでの出力ｻｲｸﾙ・ｽﾀｰﾄ。
CAZ		I/O	8ﾋﾞｯﾄ・ﾏｽﾀ・ﾄﾗﾝｻﾞｸｼｮﾝの入力ｻｲｸﾙ肯定応答。 16ﾋﾞｯﾄ・ｽﾚｰﾌﾞ・ﾄﾗﾝｻﾞｸｼｮﾝの入力書き込みまたは読み出しｽﾄﾛｰﾌﾞ。 CSRｱｸｾｽ・ﾓｰﾄﾞの出力ｻｲｸﾙ肯定応答。
WEZ		I/O	8ﾋﾞｯﾄ・ﾏｽﾀ・ﾄﾗﾝｻﾞｸｼｮﾝの出力書き込み指示。 CSRｱｸｾｽ・ﾓｰﾄﾞの入力書き込み指示。
REZ			
HCLK		O	ホスト・クロック、HCLK_SEL を用いてプログラム可能 00 = SYSCLK 01 = SYSCLK/2 10 = SYSCLK/4 11 = SYSCLK/8
AADEC[1:0]		O	非同期アドレス復号 00 = ﾏｲｸﾛｺﾝﾄﾛｰﾗ・ｴﾐｭﾚｰｼｮﾝ、CSRｱﾄﾞﾚｽ Micro_M_Reg. 復号 01 = メモリ、初期メモリ空間復号 10 = 外部CSR、ｺﾝﾄﾛｰﾗ200にない ﾚｼﾞｽﾀ空間復号 11 = 内部ＣＳＲ，ｺﾝﾄﾛｰﾗ200に あるﾚｼﾞｽﾀ空間復号
HTYPE[1:0]		I	ﾎｽﾄ・ｲﾝﾀﾌｪｰｽのﾀｲﾌﾟ 00 = 8-ﾋﾞｯﾄ・ﾏｽﾀ ｺﾝﾄﾛｰﾗ200はA(ｱﾄﾞﾚｽ)，CSZおよびWRをｱｻｰﾄする。 D(ﾃﾞｰﾀ)はWRZに応じて双方向である。 外部ｴｰｼﾞｪﾝﾄは肯定応答ﾄﾗﾝｻﾞｸｼｮﾝにCAZを用いる。 01 = 16-ﾋﾞｯﾄ・ｽﾚｰﾌﾞ
			ｲﾝﾌﾟﾗｲﾄﾞｱﾄﾞﾚｽ 自動ｼｰｹﾝｽ 10 = CSRｱｸｾｽ・ﾓｰﾄﾞ このﾓｰﾄﾞでは、ﾎｽﾄ・ｲﾝﾀﾌｪｰｽはｽﾚｰﾌﾞとして働く。読み出しおよび書き込みﾄﾗﾝｻﾞｸｼｮﾝは A, D, CSZ, CAZおよびWRZを用いて履行され、4K CSRにｱｸｾｽする。
TEVENT		I	立ち上がりｴｯｼﾞが Sensor_Prefetch_Event をﾄﾘｶﾞ。 立ち下がりｴｯｼﾞが Sensor_Transmit_Event または Manual_Transmit_Event をﾄﾘｶﾞ。
D[0:1]		I/O	Phy-linkﾃﾞｰﾀ・ﾊﾞｽのﾃﾞｰﾀ0-1は100 Mb/sでD[0:1]上に期待される。D[0]　はMSﾋﾞｯﾄである。
CTL[0:1]		I/O	Phy-link制御ﾊﾞｽの制御0-1。CTL[0]はMSﾋﾞｯﾄである。
LREQ		O	Phyへのlink要求。この出力はﾊﾞｽ要求を作るためおよびPhyﾚｼﾞｽﾀにｱｸｾｽするために用いられる。
SYSCLK		I	ｼｽﾃﾑ・ｸﾛｯｸ。この入力はPhy からの49.152Mhzである。

【００６６】次に図１７を参照すると、コードレット書き込み要求がホストとのインタフェースのため非同期モードで受信された場合の動作のタイミング図が示されている。ブロック３０１で示されているように、書き込み要求が受信される。ホスト・クロックＨＣＬＫの立ち上がりエッジから短い経過の後、ＣＳＺが「ロー」になり、出力サイクル開始／チップ選択動作を指示する。データとアドレスの両方が複号されてアドレスおよびデー

タ・バス上に置かれる。ＷＥＺ線もＣＳＺ線と共に「ロー」になり、出力書き込み動作を指示する。この期間中、ＡＡＤＥＣ線は「０」に設定される。ＣＳＺとＷＥＺが「ロー」の間、肯定応答（ＡＣＫ）が送られる。その後、書き込み応答が続く。

【００６７】次に図１８を参照すると、コードレット読み出し要求が扱われる場合の動作のタイミング図が示されている。ＣＳＺ線が短い遅れの後「ロー」になり、それに続いてＣＡＺが「ロー」になって、入力マスタ・トランザクションの入力サイクル肯定応答信号を示す。これはマスタと認められる。ＣＳＺが「ロー」になると、読み出し動作のためのアドレスがバス上に置かれ、次いでＣＡＺが「ロー」になると、データが周辺装置からバス上に置かれる。その後、このデータはＦＩＦＯ２１９に記憶され、応答として要求端に送られる。

【００６８】次に図１９を参照すると、コードレット書き込み要求がメモリ・モードで受信された場合の動作のタイミング図が示されている。これらは内部ＦＩＦＯ２１９を用いたノードの初期メモリ空間にアドレスされたブロック読み出しまたは書き込み要求である。これは、データのブロックが３つ以上のサイクルの間に「ロー」になり、「ハイ」に上がり、その後「ロー」に下がって順次処理される点でＳＲＡＭタイミングと類似している。その結果、４つのアドレスおよび関連したデータが受信され書き込まれる。

【００６９】次に図２０を参照すると、メモリ・モードでの読み出し要求を示すタイミング図が示されている。これは図１９を参照した上の説明と同様に動作するが、これは初期メモリ空間への読み出し要求でバイト・サイズのトランザクションが実行されるので、４回繰り返される点で異なる。ＦＩＦＯ２１９が一杯になった後、適当な読み出し応答が送られる。

【００７０】

【発明の効果】要約すると、本発明によれば、マイクロコントローラ・エミュレータを内蔵したリンク／トランザクション層コントローラが提供される。このマイクロコントローラ・エミュレータはリンク／トランザクション層にマイクロコントローラの動作を組み入れるのを可能にし、それにより、送信ノードは周辺装置のアドレス空間を仮想的にアドレス指定でき、別個のマイクロコントローラおよび必要なインタフェース・トランザクションを経る必要性なしに、書き込み動作中はそこにデータを送信し、また読み出し動作中はそこからデータを読み出すことができる。そのため、遠隔の送信ノードはアクセスする位置を直接制御することができ、実質的にマイクロコントローラ動作の幾つかを要求元ノードにおいて実行することができる。本発明の好適実施例を詳細に説明したが、種々の変更、置換および代替が特許請求の範囲に記載の発明の精神および範囲を逸脱することなくなされ得ることが理解されるべきである。

【図面の簡単な説明】

【図１】ＩＥＥＥ１３９４シリアル・バス・アーキテクチャを用いたシステムの全体的なブロック図。

【図２】ＩＥＥＥ１３９４バス内の種々のプロトコル層の簡略化されたブロック図。

【図３】ＦＩＦＯとインタフェースする物理層およびリンク層のより詳細なブロック図。

【図４】シリアル・バス上の非同期送信の一例を示す図。

【図５】リンク層がトランザクションをどのように扱うかを示す線図。

【図６】プライマリ・パケット・データのフォーマットを示す図。

【図７】ａは、マイクロコントローラ・エミュレータ・インタフェースを有するリンク／トランザクション層コントローラのブロック図。ｂは、周辺ユニットへのデータ・バス・インタフェースの別の実施例を示すブロック図。

【図８】リンク／トランザクション層コントローラのディジタル・カメラへの応用を示すブロック図。

【図９】リンク／トランザクション層コントローラのステレオ・オーディオ送信器での使用のための応用を示すブロック図。

【図１０】リンク／トランザクション層コントローラのステレオ・オーディオ受信器への応用を示すブロック図。

【図１１】シリアル・バスの一方側でリンク／トランザクション層コントローラに関連付けられた周辺ユニットと該周辺ユニットをリンク／トランザクション層コントローラを通して制御するために他方側の遠隔位置でシリアル・バスにインタフェースされたＣＰＵのブロック図。

【図１２】ＩＥＥＥ１３９４バス用の６４ビット・アドレッシング・マップの線図。

【図１３】リンク／トランザクション層コントローラ用のレジスタ・マップの線図。

【図１４】制御レジスタの線図。

【図１５】ホストＣＳＲアクセスのためのタイミング図。

【図１６】リンク／トランザクション層コントローラのピン接続を示す線図。

【図１７】コードレット書き込み要求を受信する動作のためのタイミング図。

【図１８】コードレット読み出し要求の受信のためのタイミング図。

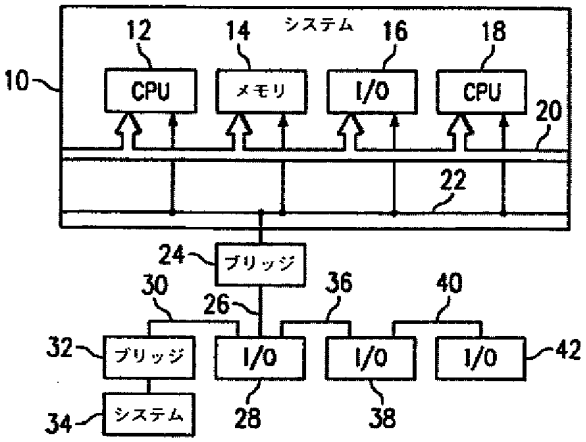
【図１９】非同期アドレス複号動作がメモリに対してなされる場合のコードレット書き込み要求の受信のためのタイミング図。

【図２０】非同期アドレス複号がメモリになされる場合のコードレット読み出し要求の受信のためのタイミング

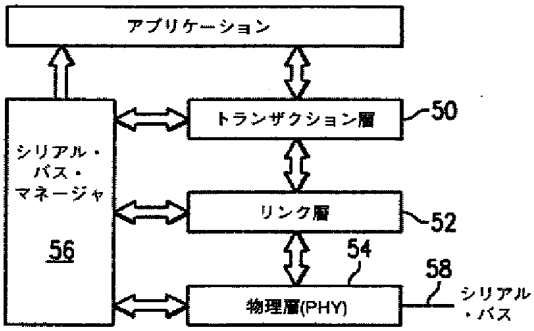
35　　36

図。
【図２１】ブロック書き込み要求の受信のためのタイミング図。
【図２２】ブロック読み出し要求の受信のためのタイミング図。
【符号の説明】
５０　トランザクション層
５２、２８６　リンク層
＊５４、２８８　物理層
５８　シリアル・バス
２００　リンク・トランザクション層コントローラ
２０９、２１０、２８４　周辺装置（ユニット）
２８０　遠隔ノード
２８２　ローカル・ノード
２９０　ＣＰＵ
＊

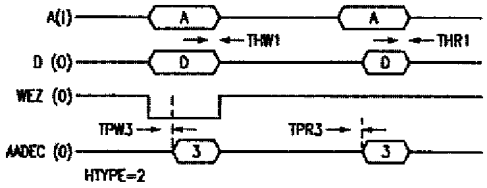
【図１】



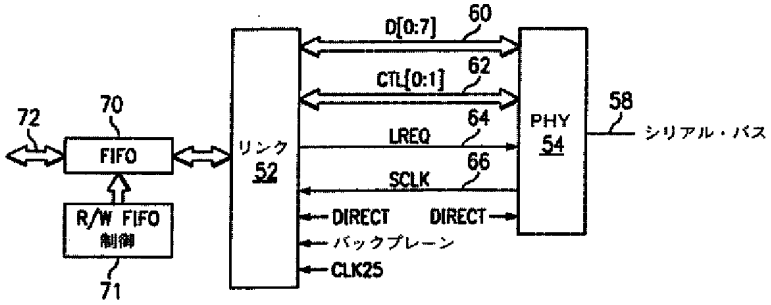
【図２】



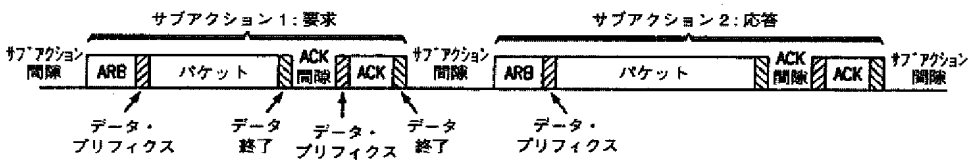
【図１５】



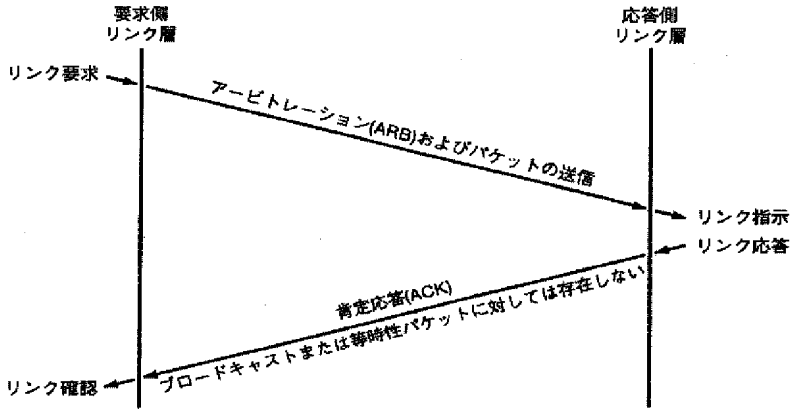
【図３】



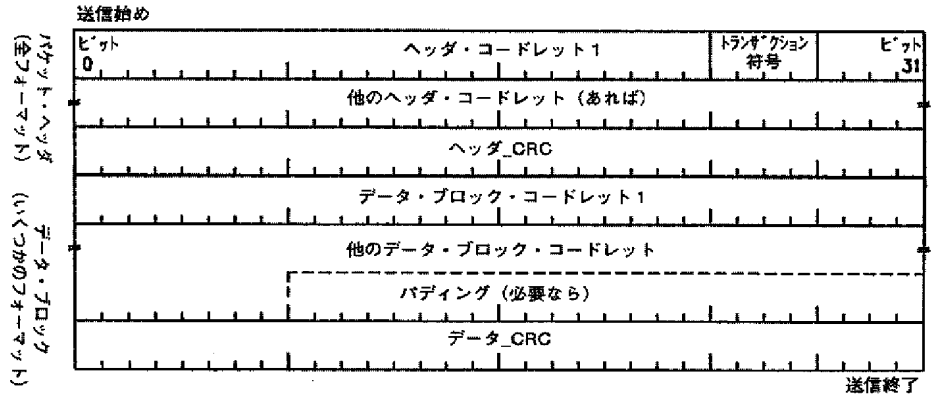
【図４】



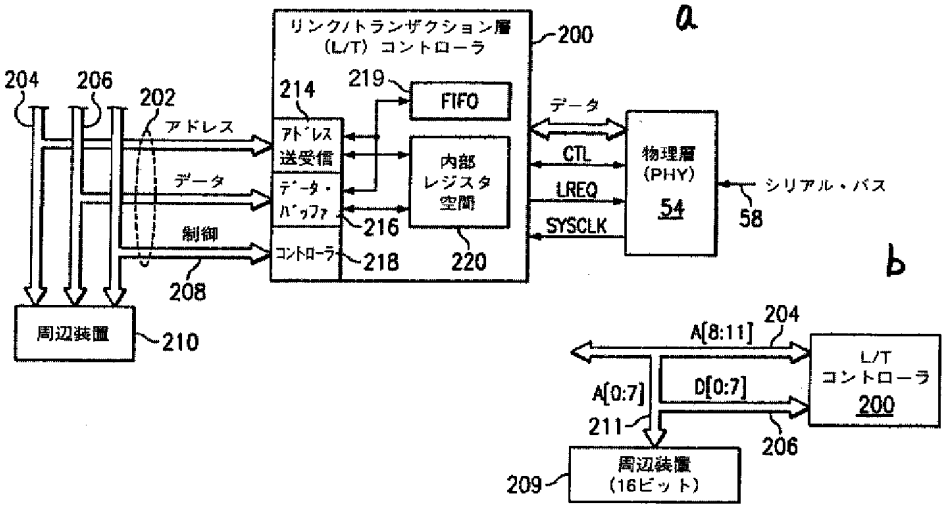
【図５】


要求側
リンク層
応答側
リンク層
リンク要求
アービトレーション(ARB)およびパケットの送信
リンク指示
リンク応答
肯定応答(ACK)
ブロードキャストまたは等時性パケットに対しては存在しない
リンク確認

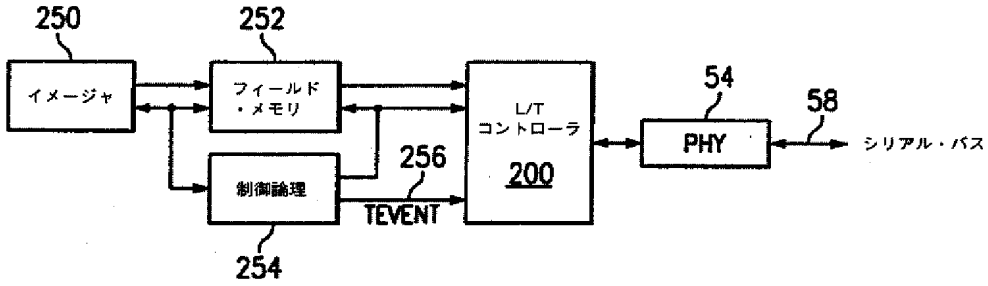
【図６】


送信始め
ビット 0
ヘッダ・コードレット１
トランザクション符号
ビット 31
他のヘッダ・コードレット（あれば）
ヘッダ_CRC
データ・ブロック・コードレット１
他のデータ・ブロック・コードレット
パディング（必要なら）
データ_CRC
送信終了
パケット・ヘッダ（全フォーマット）
データ・ブロック（いくつかのフォーマット）

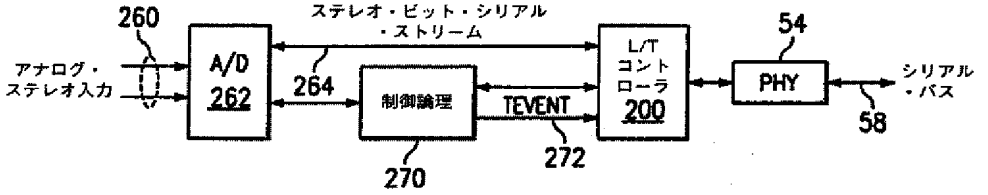
【図７】


a
リンク/トランザクション層
(L/T) コントローラ
200
214
219
FIFO
204
206
202
アドレス
データ
制御
アドレス送受信
データ・バッファ
コントローラ
216
218
内部レジスタ空間
220
208
周辺装置
210
データ
CTL
LREQ
SYSCLK
物理層
(PHY)
54
シリアル・バス
58
b
A[8:11]
204
A[0:7]
D[0:7]
211
206
L/T
コントローラ
200
209
周辺装置
(16ビット)

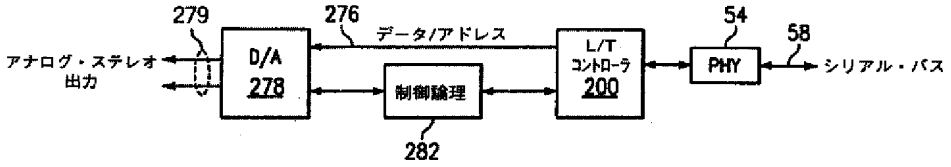
【図８】



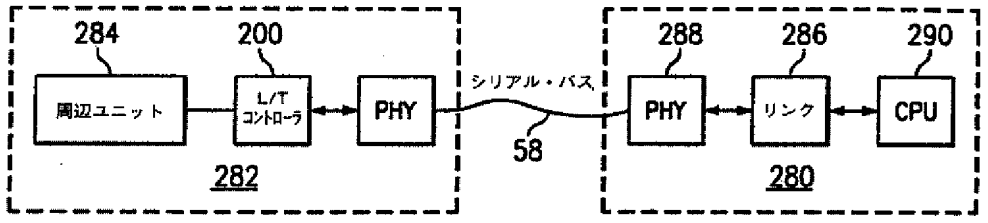
【図９】



【図１０】



【図１１】

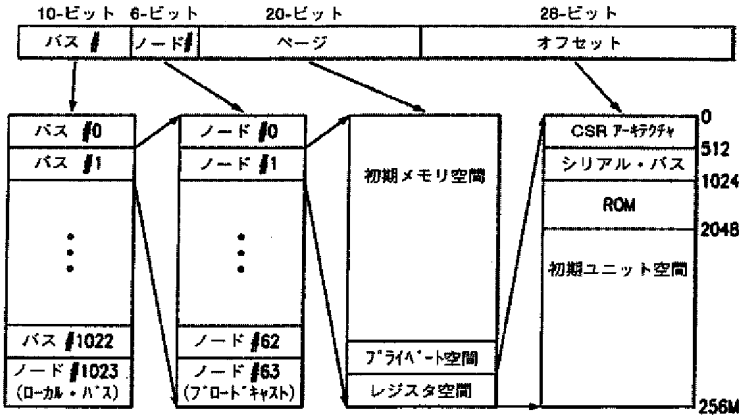


【図１４】

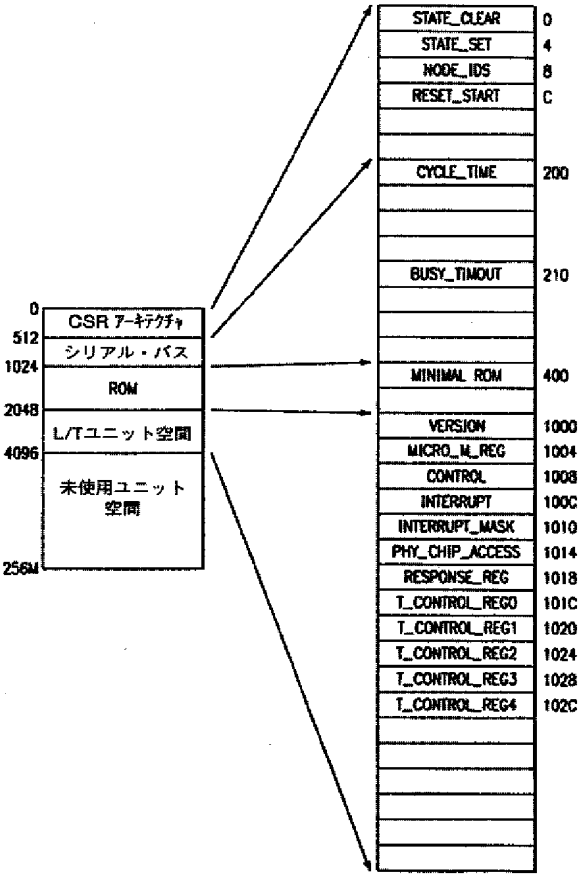
保留	NIE	SBMR	SBMS	SBIS	TL	RT	TCODE	PRI	T_CONTROL_REG0
DESTINATION_ID					DESTINATION_OFFSET_HI				T_CONTROL_REG1
DESTINATION_OFFSET_LO									T_CONTROL_REG2
DATA_LENGTH					保留				T_CONTROL_REG3

DMEN	TX_THRESHOLD		TAG	CHANNEL	RSRVD	SY	T_CONTROL_REG4

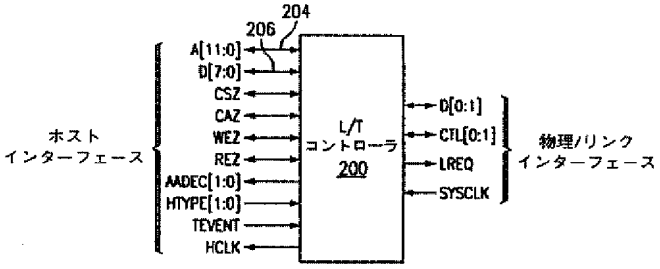
【図１２】



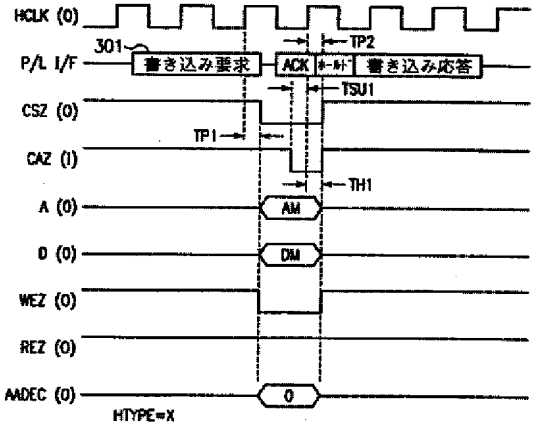
【図１３】



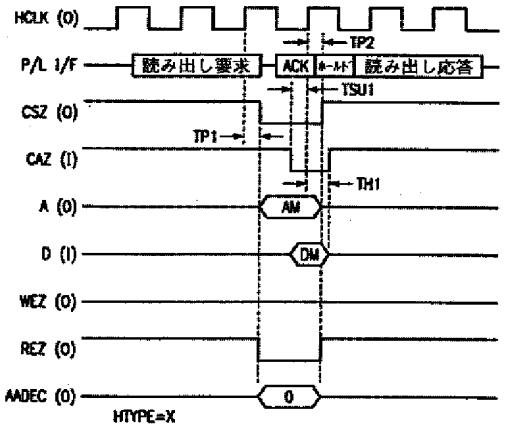
【図１６】



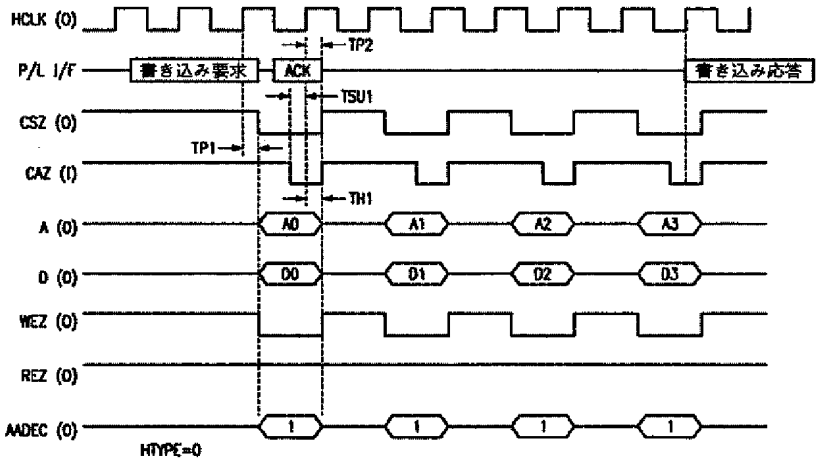
【図１７】



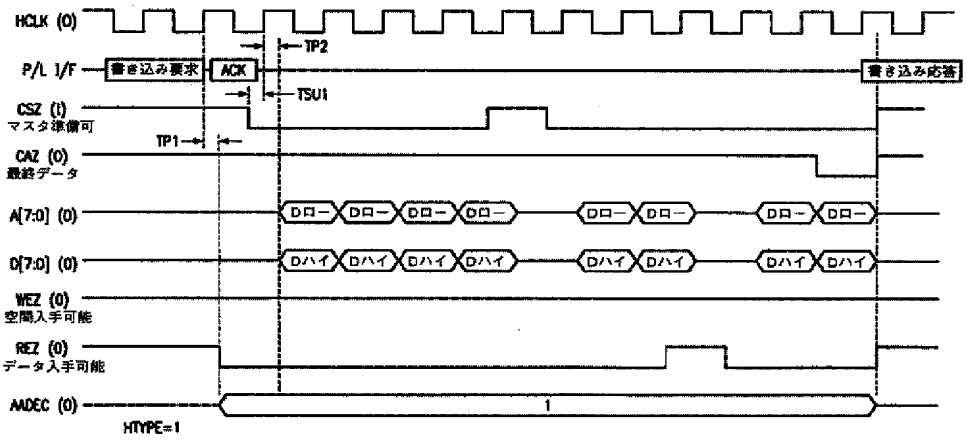
【図１８】


HCLK (0)
TP2
P/L I/F
読み出し要求
ACK
ホールド
読み出し応答
TSU1
CSZ (0)
TP1
CAZ (I)
TH1
A (0)
AM
D (I)
DM
WEZ (0)
REZ (0)
AADEC (0)
0
HTYPE=X

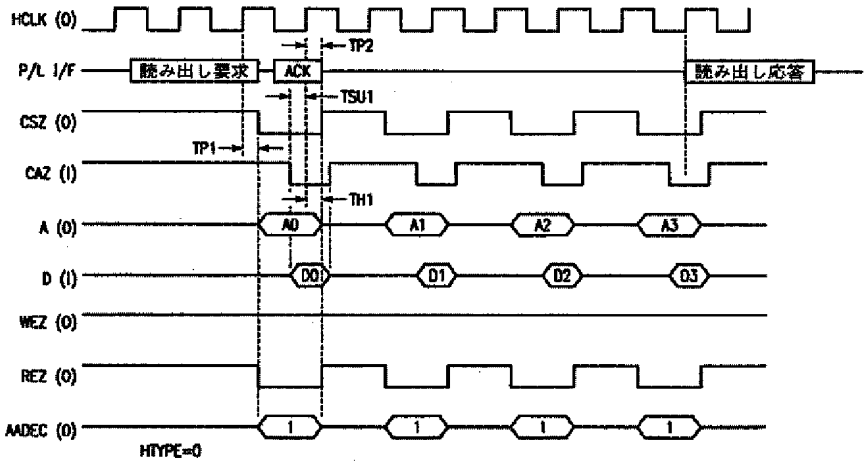
【図１９】


HCLK (0)
TP2
P/L I/F
書き込み要求
ACK
書き込み応答
TSU1
CSZ (0)
TP1
CAZ (I)
TH1
A (0)
A0
A1
A2
A3
D (0)
D0
D1
D2
D3
WEZ (0)
REZ (0)
AADEC (0)
1
HTYPE=0

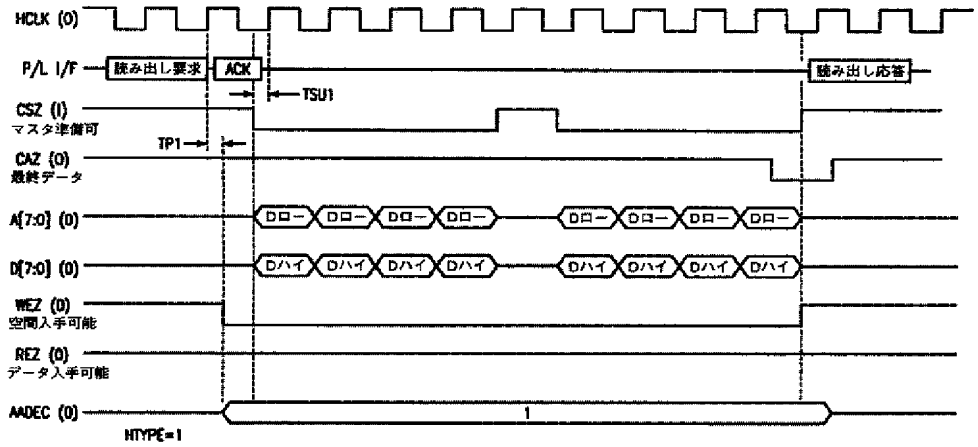
【図２１】


HCLK (0)
TP2
P/L I/F
書き込み要求
ACK
書き込み応答
TSU1
CSZ (I)
マスタ準備可
TP1
CAZ (0)
最終データ
A[7:0] (0)
D[7:0] (0)
Dハイ
WEZ (0)
空間入手可能
REZ (0)
データ入手可能
AADEC (0)
1
HTYPE=1

【図２０】



【図２２】



フロントページの続き

(72)発明者　バーク　ヘネハン
アメリカ合衆国，テキサス，ダラス，オーデリア　12516，アパートメント　ナンバー1706

【外国語明細書】

TXIN-24,054/TI-24440 PATENT
EXPRESS MAIL CERT. NO. EL021317434US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

LINK/TRANSACTION LAYER CONTROLLER WITH INTEGRAL MICROCONTROLLER EMULATION

TECHNICAL FIELD OF THE INVENTION

The present invention pertains in general to receiving data from a serial bus of the IEEE 1394 type and, more particularly, to interfacing that data to a peripheral unit in a host system without the requirement for a separate microcontroller.

BACKGROUND OF THE INVENTION

The IEEE has approved a new standard under IEEE 1394 for a high-performance serial bus cable environment that includes a network of logical nodes connected by point-to-point links called physical connections. The physical connections consist of a port on each of the nodes and a cable disposed therebetween. A node can have multiple ports, which allows a branching multi-hop interconnect. The limitations on this topology are set by the requirement for the fixed round-trip time required for the arbitration protocol. The default timing set after a bus reset is adequate for 16 cable hops, each of 4.5 meters for a total of 72 meters. The maximum number of nodes supported on a single bus is 63.

Whenever a node is added to or removed from the 1394 serial bus, a bus reset occurs and forces all nodes to a known state. After a bus reset, the tree identify (ID) process translates the general network topology into a tree, where one node is designated a root, and all the physical connections are labeled as either a parent, a child or as unconnected. The unconnected ports are labeled as "off" and do not participate any further. The tree must be acyclic, meaning no loops allowed; otherwise, the tree ID process will not be completed.

The 1394 cable environment supports multiple data rates of 98.304, 196.608, 393.216 megabits per second. The lowest speed is known as the base rate, and all ports that support a higher data rate must also support the lower data rate. Nodes capable of data rates greater than the base rate exchange speed information with its peers through its attach ports during the speed signaling phase of normal bus arbitration. If a peer node is incapable of receiving high-speed data, then data will not propagate down that path. Data will only be propagated down paths that support the higher data rate.

During data packet transmission, the source node sends a speed code, format and transaction codes, addresses of the source and destination nodes and data in a packet form. The destination field in this packet is utilized by each node's link

layer to determine if it is the recipient of the transmitted data. The maximum speed at which a data packet can be transmitted depends upon the bus topology and the data transmission speed supported by the nodes on the bus. To determine the optimum speed at which a data packet may be sent, the maximum supported speeds of the transmitting and receiving nodes, as well as the maximum speeds of any nodes connected between these nodes, must be determined. The optimum speed for data transmission is equal to the highest speed which is supported by all the nodes, which are required to participate in the transmission of the data packet.

The IEEE 1394 bus typically requires a physical layer for extracting information from the bus at a transaction/link layer for interfacing the extracted data from the bus to a host system. The host system typically comprises a host bus and a CPU. The CPU is generally given the task of extracting information from a FIFO in which data from the bus is stored. This data, after being fetched by the CPU, is then transmitted to the appropriate peripheral unit or utilized in various processing operations by the CPU. The CPU can also send information to the serial bus by first storing it in the FIFO and then providing instructions to the link/transaction layer to retrieve the information from the FIFO and transmit it to the serial bus. However, due to the fact that a separate CPU or microcontroller is required, this makes the IEEE1394 bus less attractive for small applications such as digital microphones, stereo receivers and transmitters, etc.

SUMMARY OF THE INVENTION

The present invention disclosed and claimed herein comprises a serial bus interface disposed on a local node for interfacing between a serial bus and a host system and for receiving information from the serial bus placed thereon by a remote node, and transferring this received data to the host system, and receiving information from the host system and transferring the received information to the serial bus for reception by the remote node. The interface includes a data receiver for receiving data generated by the remote node from the serial bus, and a data transmitter for transmitting data to the serial bus for receipt at the remote node. A plurality of registers are also provided, at least one of the registers addressable by the remote node for storage of received data therein. The data receiver is operable to store received data in at least one register during a read operation with a transmitter operable to transmit data to the serial bus during a Write operation. A host bus interface is provided to interface directly with a host bus on the host system, the host bus interface operable to transfer data stored in the at least one register to the host bus during a Write operation when the data is received and stored in the at least one register. It is also operable to retrieve data from the host bus during a Read operation.

In another aspect of the present invention the interface includes a standard register space, wherein at least one register occupies a portion of the standard register space. Select ones of the plurality of registers are dedicated to storage of standard bus information and the remote node can directly access this information. The data receiver is operable to recognize a request for access to one of the plurality of registers and the data transmitters are then operable to transmit the contents thereof when addressed. Select ones of the plurality of registers contain configuration registers for storing configuration information therein that define the operation of the serial bus interface. This allows the remote node to program the operation of the serial bus interface by accessing one of the configuration registers.

In a further aspect of the present invention, data is received in data packets and transmitted in data packets. These data packets comprise information necessary to identify the transmitting node on the serial bus and the content of the data packet in addition to information identifying the remote node designated to receive the packet. Each data reception or data transmission operation is proceeded by a data request, a Write request or Read request, respectively, from the remote node, which requests are contained in the received data packet. The received data packet associated with a Write request contains the data associated therewith, which data is stored in at least one register upon receipt thereof. The host interface recognizes Write requests and then transfers the data stored the at least one register to the host bus. During a Read request, the host interface recognizes the Read request and access the data from the host for transfer to remote node with data transfer.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following description taken in conjunction with the accompanying Drawings in which:

FIGURE 1 illustrates an overall block diagram of a system utilizing the IEEE 1394 serial bus architecture;

FIGURE 2 illustrates a simplified block diagram of the various protocol layers in the IEEE 1394 bus;

FIGURE 3 illustrates a more detailed block diagram of the physical and link layers which interface with the FIFO;

FIGURE 4 illustrates an example of an asynchronous transmission over the serial bus;

FIGURE 5 illustrates a diagrammatic view of how the link layer services a transaction;

FIGURE 6 illustrates a primary packet data format;

FIGURE 7 illustrates a block diagram of the link/transaction layer controller with the microcontroller emulator interface;

FIGURE 7a illustrates an alternate embodiment of the data bus interface to the peripheral unit;

FIGURE 8 illustrates an application of the link/transaction layer controller for a digital camera;

FIGURE 9 illustrates a block diagram of an application of the controller for use with a stereo audio transmitter;

FIGURE 10 illustrates a block diagram of an application of the controller for a stereo audio receiver;

FIGURE 11 illustrates a block diagram of a peripheral unit associated with the link/transaction layer controller on one side of the serial bus and a CPU interfaced with the serial bus on the other side at a remote location for controlling the peripheral unit through the link/transaction controller;

FIGURE 12 illustrates a diagrammatic view of the 64-bit addressing map for the IEEE 1394 bus;

FIGURE 13 illustrates a register map for the link/transaction layer controller;

FIGURE 14 illustrates a diagrammatic view of the control register;

FIGURE 15 illustrates a timing diagram for the host CSR access;

FIGURE 16 illustrates a diagrammatic view of the pin connections for the link/transaction layer controller;

FIGURE 17 illustrates a timing diagram for the operation of receiving a quadlet write request;

FIGURE 18 illustrates a timing diagram for the receipt of a quadlet read request;

FIGURE 19 illustrates a timing diagram for the receipt of a quadlet write request wherein the asynchronous address decode operation is for memory;

FIGURE 20 illustrates a timing diagram for the receipt of a quadlet read request wherein the asynchronous address decode is to the memory;

FIGURE 21 illustrates a timing diagram for the receipt of a block write request; and

FIGURE 22 illustrates a timing diagram for the receipt of a block read request.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to FIGURE 1, there is illustrated a block diagram of a system utilizing the serial bus architecture that is defined as the IEEE 1394 serial bus. This is defined in the "IEEE Standard for a High-Performance Serial Bus," IEEE STD 1394-1995, which is incorporated herein by reference. A module 10 is illustrated, which module 10 includes a CPU 12, a memory 14, an input/output (I/O) 16 and a CPU 18. The CPU 12, memory 14, I/O 16 and CPU 18 all comprise units within the system. Each of the units 12-18 interfaces with a parallel bus 20, which is a system bus that is indigenous to the module 10. In addition, each of the units 12-18 interfaces with a serial bus 22 which is referred to as a "backplane." The serial bus 22 operates in accordance with the IEEE 1394 standard and is also interfaced external to the system with a bridge 24. The bridge 24 and the module 10 each comprise logical nodes on the serial bus. In general, the serial bus architecture is defined in terms of logical nodes, the node being an addressable entity. Each of those can be independently reset and identified, and more than one node may reside on a single module, and more than one unit may reside in a single node. A node is therefore a logical addressing concept wherein a module is a physical device which can consist of more than one node that share a physical interface. The address space of a single node can be directly mapped to one or more units. A unit can be a logical entity, such as a disk controller, memory, CPU, etc. Within a given unit, there may be multiple subunits, which can be accessed through independent control registers or uniquely addressed with direct memory access (DMA) command sequences.

Referring further to FIGURE 1, it can be seen that there are two environments, one within the module 10 utilizing the backplane 22, referred to as the "backplane environment," and the other being a "cable environment." The nodes interfacing with the cable environment have "ports" associated therewith. The bridge node 24 is such a node which interfaces on one side to the backplane serial bus 22 and on the other side to a cable 26 which interfaces to a single I/O node 28 through one port therein. The I/O node 28 has two other ports, one of which is connected through a cable serial bus 30 to a bridge node 32. The bridge node 32 is similar to the bridge

node 24 in that it interfaces with another system 34, this being a module. The system node 34 can be substantially identical to the system 10, or any other type of system employing a backplane. The third port of the I/O node 28 interfaces through a cable serial bus 36 to one port of an I/O node 38, the other port thereof interfaced through a cable serial bus 40 to an I/O node 42.

The cable environment in general is a physical topology that provides a noncyclic network with finite branches and extent. The medium consists of two conductor pairs for signals and one pair for power and ground that connect ports on different nodes. Each port consists of terminators, transceivers, and simple logic. The cable and ports act as bus repeaters between the nodes to simulate a single logical bus. The backplane environment, by comparison, comprises a multidrop bus. This consists of two single-ended conductors running the length of the backplane in the module. Connectors distributed along the bus allow nodes to "plug into" the bus. This system makes use of wired-OR logic that allows all nodes to assert the bus.

Referring now to FIGURE 2, there is illustrated a block diagram of the serial bus protocol. The serial bus protocol is comprised of three stack layers, a transaction layer 50, a link layer 52 and a physical layer 54 labeled "PHY." The transaction layer defines a complete response-response protocol to perform the bus transactions required to support the CSR architecture (control and status registers). This provides operations of read, write and lock. The link layer 52 provides an acknowledge datagram (a one-way data transfer with confirmation of request) service to the transaction layer 50. It provides addressing, data checking, and data framing for packet transmission and reception. The link layer 52 also provides an isochronous data transfer service directly to the application, including the generation of a "cycle" signal utilized for timing and synchronization. One link layer transfer is called a "subaction."

The physical layer 54 provides three major functions. It translates the logical symbols utilized by the link layer 52 into electrical signals on the different serial bus media. It guarantees that only one node at a time is sending data over the bus by providing an arbitration service. It also defines the mechanical interfaces for the

serial bus. For each environment, there is provided a different physical layer, the cable and backplane environments. The cable physical layer also provides a data resynch and repeat service and automatic bus initialization.

In addition to the three layers, there is also provided a serial bus management block 56 that provides the basic control functions and standard CSRs needed to control nodes or to manage bus resources. This includes a number of components, a bus manager component which is only active at a single node that exercises management responsibilities over an entire bus, a node controller component, and an isochronous resource manager that centralizes the services needed to allocate data with another isochronous resource. An isochronous resource is a resource having the characteristics of a time-scale or signal that has time intervals between consecutive significant instances with either the same duration or durations that are integral multiples of the shortest duration. For the purposes of the present invention, the physical layer interfacing with the serial bus 58 and the link layer 52 will be interfaced with a receive buffer (not shown).

Referring now to FIGURE 3, there is illustrated a block diagram of the interface between the physical layer 54 and the link layer 52. The physical layer 54 interfaces with the serial bus 58 and is operable to receive data therefrom. Data is passed to and from the link layer 52 through an 8-bit bi-directional data bus 60. Two control bits are passed between the physical layer 54 and the link layer 52 over a control bus 62. A link request is transferred to the physical 54 from the link layer 52 through a request line 64, but with a system clock signal, SCLK, transferred from the physical layer 54 to the link layer 52, the physical layer 54 recovering this clock.

Hereinafter, data rates are referred to in multiples of 98.304 Mbit/s. The interface provided in the IEEE 1394 in the cable environment will support the following data rates: 100 Mbit/s, 200 Mbit/s and 400 Mbit/s. The backplane environment will support 25 Mbit/s and 50 Mbit/s. These rates are actually "bit" rates, independent of the encoding scheme. The actual clock rate in a redundant encoding scheme is referred to as a "baud" rate and is independent of the clock rate of this interface.

The physical layer 54 has control over the bi-directional pins for transferring the data and the control bits. The link layer 52 only drives these pins when control is transferred to it by the physical layer 54. The link performs all unsolicited activities through a dedicated request pin on line 64. The possible actions that may occur on the interface are categorized as transmit, receive, status and request. The SCLK is driven by the physical layer 54 and is generally synched to the serial bus clock at a rate of 49.152 MHz. There is provided a backplane input on the link layer 52 which, if set high, indicates that the physical layer is in a backplane environment. Another input, a Clk25 input, when set high, forces the SCLK output from the physical layer 54 on the line 66 to a value of 24.576 MHz.

When data is carried between the two chips, the width of the data bus 60 depends on the maximum speed of the connected physical layer 54, two bits for every 100 Mbit/s. Therefore, packet data for a 100 Mbit/s transmit utilizes D[0:1], 200 Mbit/s transfers utilize D[0:3], and 400 Mbit/s transfers utilize the full D[0:7]. The unused D[n] signals are driven low. The control bus 62 always carries two bits. Whenever control is transferred between the physical layer 54 and the link layer 52, the side surrendering control always drives the control and data pins to a logic "0" level for one clocks before tri-stating its output buffers.

As noted above, there are four basic operations that may occur in the interface: request, status, transmit and receive. To request the bus or access a register in the physical layer 54, the link layer 52 sends a short serial stream to the physical layer 54 on the request pin 64. When the physical layer 54 has status information to transfer to the link layer 52, it will initiate a status transfer. The physical layer 54 will wait until the interface is idle to perform this transfer, and it will initiate the transfer by asserting a status bit on the control bus 62. It will also provide at the same time the first two bits of status information on D[0:1]. When the link requests access to the serial bus through the request line 64, the physical layer 54 arbitrates for access to the serial bus 58. When it wins the arbitration, it will then grant the bus to the link layer 52 by asserting "transmit" on the control bus 62 for one cycle of the SCLK. It will then be idle for a single cycle. After sampling the "transmit" state from physical

layer 54, the link layer 52 will then take over control of the interface by asserting either a "hold" or a "transmit" on the control bus 62. During a receive operation, whenever the physical layer 54 sees a "data-on" state on the serial bus 58, it initiates a receive operation by asserting "receive" on the control bus 62 and a logic "1" on each of the data pins. Physical layer 54 then indicates the start of a packet by placing the speed code on the data pins. For 100 Mbit/s, the data bits will be "00xxxxxx," and for 200 Mbit/s, it will be "0100xxxx," with 400 Mbit/s being "01010000," the value "x" being a non operation.

The link layer 52 will interface with a buffer in the form of a FIFO 70, which is controlled by a read/write FIFO control block 71 that defines the position of the read and write pointers and all accesses in and out of the FIFO. The other side of the FIFO 70 is interfaced with the host bus 72, which host bus 72 is a 32-bit bus.

Referring now to FIGURE 4, there is illustrated a subaction in the link layer 52 for an asynchronous transmission of a packet. This subaction is in the form of a request and a response. There is provided an arbitration sequence which is transmitted by a node that wishes to transmit a packet, this being transmitted to the physical layer 54 to gain control of the bus 58. The physical layer 54 may then respond immediately if it already controls the bus. This is followed by a data packet transmission which, for asynchronous subactions, involves the source node sending a data prefix signal (including a speed code, if needed), addresses of the source node and destination nodes, a transaction code, a transaction label, a retry code, data, one or two cyclic redundancy checks (CRCs), and a packet termination (either another data prefix or a data end signal). This is all followed by an acknowledgment field wherein a uniquely addressed destination returns a code indicating to the transmitting node the action taken by the packet receiver. Each of these asynchronous subactions is separated by periods of idle bus called "subaction gaps." This gap is disposed between the packet transmission and acknowledgment reception. This "ack-gap" is of varying lengths depending upon where the receiver is on the bus with respect to the senders of the link request and acknowledgment (ack). However, the maximum length of the ack-gap is sufficiently shorter than a subaction gap to ensure that other

nodes on the bus will not begin arbitration before the acknowledgment has been received.

Referring now to FIGURE 5, there is illustrated a diagrammatic view of the manner in which the link layer 52 services a request. As noted above, the link layer 52 utilizes the request, indication, response and confirmation service primitives. The request primitive is utilized by a link requestor to transfer the packet to a link responder. An indication primitive indicates the reception of a packet by a link responder. A response primitive indicates the transmission of an acknowledgment by a link responder, and the confirmation primitive indicates the reception of the acknowledgment by the link requestor. Once the link request has been made, the system goes through an arbitration and packet transmission to the receiving node, which then provides a response back in the form of an acknowledgment to the requesting link layer, which will then confirm transmission.

Referring now to FIGURE 6, there is illustrated a register map for a packet that is transmitted. The packet is configured with a header that contains a plurality of quadlets. Typically, the first quadlet will contain the physical ID and the last quadlet will contain the header CRC. The header packet is followed by a data block which consists of a plurality of data quadlets with the last quadlet being a data CRC quadlet. The packet header in the first quadlet thereof contains a transaction code which defines the packet type of a primary packet. The transaction packet code specifies the packet format and the type of transaction that shall be performed. This could be such things as a write request for a data quadlet, a write request for a data block, read requests for data quadlets and data blocks, and read responses for data quadlets and data blocks. The asynchronous data packet noted above with respect to FIGURE 4 is a primary data packet.

Referring now to FIGURE 7, there is illustrated a block diagram of the preferred embodiment of the present invention. As described hereinabove, the physical layer 54 is interfaced with both a link layer 52 and a transaction layer 50, this being illustrated in FIGURE 2. In the preferred embodiment of the present invention, there is provided a link/transaction layer controller 200 which controller 200 is

operable to interface between the physical layer and a set of host system buses 202 comprised of an address 204, a data bus 206 and a control bus 208. The host bus 202 is operable to interface with a peripheral device 210. The controller 200 is configured such that it provides all the functions of the link layer 52, the transaction layer 50, and the serial bus 56 of FIGURE 2, in addition to essentially emulating the operation of a microcontroller. Therefore, as will be described hereinbelow, the controller 200 can directly interface between the serial bus 58 and the physical device 200 without the requirement of a separate microprocessor or microcontroller.

In addition to all of the functionality of the link layer 52 and the transaction layer 50, there is provided an interface for interfacing with the host bus 202. The controller 200 is operable with the use of an address block 214 to transmit addresses to the address bus 204 and to receive from the address bus 204 addresses. Data can be transmitted to and from the data bus 206 through a data buffer 216 and control information can be transmitted to and from the control bus 208 through a controller 218. Internal to the controller 200 is an internal register space 220, which will be described in more detail hereinbelow and which is operable to define the controller 200 in a conventional address space such as an IEEE 1212 CSR address space. The internal register space 220 is internal to the controller 200 and is accessible by systems at other nodes to allow Write and Read requests directly to these registers. An internal FIFO 219 is provided, which basically functions in conjunction with the data buffer 216 to buffer data transfers between the host side of the system and the serial bus side.

In operation, a CPU (not shown) at another location on the serial bus can access the internal registers 220 to determine what type of device resides at this particular node. Once it recognizes that this node has the specific architecture that allows direct access to the peripheral devices, then data transfer can be affected in accordance with the present invention. If information in an internal register within the controller is desired by the remote location, the data packet from the transmitting device will be arranged such that the data is comprised of both address information and data information which will be recognized by the controller 200. This will be recognized as not only an address but also a request for a Write or Read operation,

which operation can then be performed. If the transmitting node desires to access peripheral device 210, it need only transmit sufficient information to the controller 200 in the form of address and data information to instruct the controller 200 to basically transfer the received address to the address bus 204 and then transmit the received data to the peripheral device 210 over the data bus 206. This is due to the fact that the peripheral unit occupies a specific portion of the address space of the controller 200 and, when such address is received, the controller will immediately transfer this address to the address bus 204 for the host side and simultaneously place the data on the host data bus, in addition to generating the appropriate Read or Write commands for the host system. This is to be compared with a conventional transaction which requires the data to be first stored in a FIFO and then an interrupt generated by the transaction layer 50 to an associated microcontroller. The microcontroller will then act upon this interrupt and retrieve the appropriate information from the FIFO and perform the subsequent processing. The system of the present invention, by comparison, directly interfaces with the system bus 202 to enable the controller 200 to autonomously receive and transmit both asynchronous and isochronous data.

In an alternate embodiment of the present invention, illustrated in Figure 7a, a portion of the address bus 204 and the data bus 206 from the controller 200 are combined. In the preferred embodiment, the data bus 206 is an 8-bit data bus and address bus 204 is a 12-bit bus. By utilizing eight bits of the address bus 204 and all eight bits of the data bus 206, a new data bus 211 can be realized which is a 16-bit data bus. This is input to a 16-bit peripheral unit 209 for allowing 16-bi data to be transferred therebetween. This is a unique mode that allows 16-bit data to be transferred in certain applications. This is merely due to the fact that a limited number of pins are provided on the chip. Additionally, it can be seen that the data is transferred to the 16-bit data bus 211 via the seven least significant bits of the address bus 207 and all of the data bits of the data bus 206 by the fact that the controller 200 recognizes that the remote node is sending information that is to be output on the address and data buses. It is important to note that the controller 200 is not actually performing the addressing operation and is making no decisions as to what is on the address bus 204 or the data bus 206; rather, it is merely transferring the data and address information to these two buses as a result of the remote node sending the

information to the controller 200 in a particular manner. It is only important that the remote node recognize how to get the information to the address bus 204 and data bus 206. As will be described in more detail hereinbelow, this merely requires the remote node to transmit the information to the specific internal register which will result in the controller 200 automatically outputting this information onto the appropriate buses. Since the remote node has knowledge of the type of device that it is transmitting information to, it will be aware of the fact that the information transmitted to the controller 200 is automatically transferred directly to the address bus 204 and the data bus 206.

Referring further to FIGURE 7, the data transferred between the host or system bus 202 and the serial bus 58 is performed in one of two ways. In the first way, data is transferred to an internal register and immediately transferred to the address transmit/receive block 214 and data buffer 216 and immediately placed on data bus 206 and address bus 204. In this manner, a remote node can transfer information directly to the system bus 202. In another mode, the data is transferred to the internal FIFO 219 for storage therein and later transfer to the peripheral device 210. In a transmit operation, the data is received from the host or system bus 202 and stored in the FIFO 219 in a store-and-forward manner. This will later be transmitted on the serial bus.

Referring now to FIGURE 8, there is illustrated an application of the controller 200 in a digital camera. The digital camera includes an imager 250 and a field memory 252 which interfaces to the imager 250 for receiving data therefrom. The camera has associated therewith a control logic block 254 which is operable, upon data being transmitted thereto, to generate a transmission event signal TEVENT on a line 256 to the controller 200. The controller 200, upon receiving this signal, will then access data in the field memory 252 to transmit this information directly to the serial bus 58 and to a remote location. As noted hereinabove, internal registers 220 are provided which can be programmed to determine how this operation will be carried out and to which node it may be transmitted.

Referring now to FIGURE 9, there is illustrated another application of the present invention, that for an audio transmitter. The analog input signal is a stereo

input signal on lines 260 which is input to an analog/digital converter 262. The A/D converter 262 is of the type TLC320AD57C, manufactured by Texas Instruments. This will generate a stereo bit-serial stream on a line 264 to the data input of the controller 200. Control logic in a block 270 is utilized to generate on a line 272 a transmission event signal to indicate that information is being transmitted to the controller 200. The controller 200 with then interface with the control logic block 270 to control the generation of the data and transmission to the controller 200 for storage in the FIFO 219, which controller 200 will then transmit this data to the serial bus 58.

Referring now to FIGURE 10, there is illustrated another application of the controller 200, wherein the controller 200 is utilized as an audio receiver. The controller 200 will receive data from the serial bus 58 in a continuous manner. This data will continually be decoded, stored in the FIFO 219 and transmitted on data/address lines 276 to a digital/analog converter 278 for output as an analog stereo output signal on lines 279. A control logic block 282 is provided for interfacing with the controller 200. The data/address information is transmitted with 8 bits of data and 8 bits of address information.

Referring now to FIGURE 11, there is illustrated a block diagram of a general application wherein a remote transmitting node 280 is operable to transmit information to a local node 282. The local node 282 is configured with the controller 200 described hereinabove and a peripheral unit 284, to which information is to be transmitted from the remote node 280. The remote node 280 has associated therewith a link layer 286 and a transaction layer (not shown) in addition to a physical layer 288. Therefore, the physical layer 288 and the link layer 286 allow the remote node 280 to communicate over the serial bus 58 with the local node 282 and send data packets thereto and receive data packets therefrom. A CPU 290 is provided at the remote node 280 for communicating with the peripheral unit 284 at the local node 282. As described hereinabove, the CPU 290 is operable to communicate with a plurality of nodes on the bus 58. To communicate with the node 282, it first must know what type of node exists at the local node 282. To do this, it will access a known register location in the address space of the controller 200, this being a typical CSR 1212

address space. The local node 282 is configured with this address space; however, all other nodes are also configured with the CSR 1212 address space. When accessing the specific address location in the controller 200, the CPU 290 need not go to another CPU on the local node but, rather, goes directly to an internal register to the controller 200 when such request is made. The controller 200 will then transmit to the serial bus 58 the requested information. Once the CPU 290 recognizes that the controller 200 exists at the local node 282, it will then generate its Read and Write requests in a slightly different manner than normal. It will transmit them knowing that it can virtually make a direct access to the address space of the peripheral unit 284 without having to go through a FIFO, this being described hereinbelow. This in effect transfers some of the functionality of a CPU that would normally reside at the local node to the remote node. Further, the CPU 290 at the remote node can also transmit configuration instructions directly to the internal registers of the controller 200 to configure these registers.

To allow a CPU 290 at the remote node to program the controller 200 at the local node, a plurality of configuration registers are mapped into the address space of the controller 200. These configuration registers are immediately accessible from the host interface. In general, the IEEE 1394 bus protocol uses a 64-bit fixed addressing scheme. This 64-bit address consists of a 10-bit bus number, 6-bit node number, 20-bit page address and 28-bit offset. There are 3 addressable memory spaces for each node. They are memory, private and register space. The addresses for each space are set forth in *Table 1*:

Table 1

Memory	Bus_Node_00000_0000000 Bus_Node_FFFFD_FFFFFFF
Private	Bus_Node_FFFFE_0000000 Bus_Node_FFFFE_FFFFFFF
Register	Bus_Node_FFFFF_0000000 Bus_Node_FFFFF_FFFFFFF

The address construction for the 64-bit addressing scheme is constructed in accordance with the diagram of FIGURE 12. In FIGURE 12, it can be seen that it is necessary to send sufficient information to the physical layer for the physical layer to extract the information from the serial bus 58. Once stripped, it need not transmit all of this

information to the link/transaction controller 200. It need only transmit the page address and the 28-bit offset information, this 28-bit offset information defining the internal register space.

Referring now to FIGURE 13, there is illustrated a diagrammatic view of the address space for the internal register space for a given controller 200. To make the controller 200 compliant with the IEEE 1212 CSR address space, it is necessary to ensure that certain information is at particular locations within the address space. Therefore, the CSR address space will have from locations 0 - 512 information as to the CSR architecture, from locations 512 - 1024, information regarding the serial bus, and from locations 1024 - 2048, Read Only Memory locations, these not necessarily being changed, from locations 2048 - 4096, information regarding the particular controller unit itself, and the remaining locations 4096 - 256M reserved as unused unit space.

Not all of the memory locations within a given region are required to provide a functioning part. Therefore, only the minimal information necessary is stored. For example, with respect to the CSR address space architecture, only 4 registers are provided, that for the STATE_CLEAR, STATE_SET, NODE_IDS and RESET_START. With respect to the serial bus region, there is only provided information regarding CYCLE_TIME and BUSY_TIMOUT in two registers. With respect to information that would be considered to be Read Only Memory information, this is stored in a location entitled MINIMAL_ROM. This generally is the information regarding the manufacturer, the part and the type associated with the part. In the locations associated with the unit space, there are stored such things as versions, types of interrupts, types of interrupt masks and various chip access information for the physical layer. Additionally, there are also provided a plurality of transmit control registers labeled T_CONTROL_REG0-4. These registers will be described in more detail hereinbelow.

Whenever the controller 200 receives a request from the serial bus 58, it decodes the entire 64-bit address. If the decoded address constitutes an address in the CSR register space, and the lower 12-bits point to an on-chip register, i.e., the registers

noted in FIGURE 13 as having information stored therein, then that register is accessed from the serial bus 58. It should be noted that there are a large number of locations which are reserved for the CSR 1212 register space which are not occupied. The reason for this is that only a minimal set of registers are provided. If, for some reason, the instructions or information normally found in the unpopulated portion of the address space, i.e., that not included in the controller 200, were required to operate from the remote node, then external registers at the local node occupying that address space would be required. Whenever a Write operation is directed toward an on-chip register at the local node controller 200, these Writes are echoed to the host interface if an internal echo bit, CSR_WR_ECHO, is set. The host interface address (A[11:0]) is simply the 12 LSBs of the 64-bit serial bus address. A special case of accessing the on-chip registers is that of accessing the Micro_M_Reg , which will be described hereinbelow. If the register space that is decoded in the lower 12-bits point to a register not on-chip, then the controller 200 will recognize this and both Writes and Reads will performed on the host interface, assuming that the CSR 1212 is implemented by the application. It should be understood that the CSR 1212 register space is only one standard and that any other register space could be utilized, as long as the remote node is aware of the particular standard being utilized by the controller 200 at any node. The CSR 1212 is utilized in the present application to make the parts compliant with a well known standard.

In general, the type of host accommodated by the chip is defined by external hardware pins labeled HTYP, which define the host type. If the host type is set to an 8-bit master, the system can emulate a microcontroller and interface the host system. If the host type is set to only CSR access mode, only the internal CSRs between locations 0 - 4096 are accessible.

There are four fundamental modes of operation for the controller 200. They are:

1. Asynchronous receive;
2. Asynchronous transmit;
3. Isochronous transmit; and
4. Host CSR access.

A summary of the transaction types supported and the mapping to the host interface transactions is set forth in *Table 2*.

Table 2: Transaction Summary

HTYPE	Received 1394 Transaction	Host Interface Transaction	ack	Transmitted 1394 Transaction	AADEC
0	Quadlet write request to Micro_M_Reg dest_addr = Node_id, FFFFF,Micro_M_Reg	8-bit master byte write A[11:0] = quadlet_data[12:23] D[7:0] = quadlet_data[24:31]	1 2 1	If Write_Resp_En is not set, no write response. If Write_Resp_En is set, Write response on CAZ Write response is concatenated if Write_Resp_Cat is set.	0
1,2	Quadlet write request to Micro_M_Reg dest_addr = Node_id, FFFFF,Micro_M_Reg	None	4	None	0
0	Quadlet read request to Micro_M_Reg dest_addr = Node_id, FFFFF,Micro_M_Reg	8-bit master byte read A[11:0] = quadlet_data[12:23] response quadlet_data[24:31] = D[7:0]	2 1	Read response on CAZ. Read response is concatenated if Read_Resp_Cat is set.	0
1,2	Quadlet read request to Micro_M_Reg dest_addr = Node_id, FFFFF,Micro_M_Reg	None	4	None	0
0	Quadlet write request to Initial Memory Space Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	Four 8-bit master byte writes (order is 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31]	1 2 1	If Write_Resp_En is not set, no write response. If Write_Resp_En is set, Write response on last CAZ Write response is concatenated if Write_Resp_Cat is set.	1
1	Quadlet write request to Initial Memory Space Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	16-bit slave mode First Access A[7:0] = quadlet_data[0:7] D[7:0] = quadlet_data[8:15] Second Access A[7:0] = quadlet_data[16:23] D[7:0] = quadlet_data[24:31]	1 2 1	If Write_Resp_En is not set, no write response. If Write_Resp_En is set, Write response on last CAZ Write response is concatenated if Write_Resp_Cat is set.	1
1	Block write request to Initial Memory Space Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	16-bit slave mode, data read from host interface until CAZ indicates last data. First Access A[7:0] = quadlet_data[0:7] D[7:0] = quadlet_data[8:15] Second Access A[7:0] = quadlet_data[16:23] D[7:0] = quadlet_data[24:31] Subsequent Accesses A[7:0] = next quadlet_data[0:7] D[7:0] = next quadlet_data[8:15]	1 2 1	If Write_Resp_En is not set, no write response. If Write_Resp_En is set, Write response on last CAZ Write response is concatenated if Write_Resp_Cat is set.	1
2	Quadlet or block write request to Initial Memory Space Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	None	4	None	1
0	Quadlet read request to Initial Memory Space Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	Four 8-bit master byte reads (order is 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31]	2 1	Read response on last CAZ. Read response is concatenated if Read_Resp_Cat is set.	1
1	Quadlet read request to Initial Memory Space Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	16-bit slave mode First Access A[7:0] = quadlet_data[0:7] D[7:0] = quadlet_data[8:15] Second Access A[7:0] = quadlet_data[16:23] D[7:0] = quadlet_data[24:31]	2 1	Read response on last CAZ. Read response is concatenated if Read_Resp_Cat is set.	1
1	Block read request to Initial Memory Space Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFE, 0000000	16-bit slave mode, data read from host interface until CAZ indicates last data. First Access A[7:0] = quadlet_data[0:7] D[7:0] = quadlet_data[8:15] Second Access A[7:0] = quadlet_data[16:23] D[7:0] = quadlet_data[24:31] Subsequent Accesses A[7:0] = next quadlet_data[0:7]	1 2 1	If Write_Resp_En is not set, no write response. If Write_Resp_En is set, Write response on last CAZ Write response is concatenated if Write_Resp_Cat is set.	1

		D[7:0] = next quadlet_data[8:15]			
2	Quadlet or block read request to Initial Memory Space Node_id, 00000, 0000000 <= dest_addr < Node_id, FFFFF, 0000000	None	4	None	1
0	Quadlet write request to External CSR (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) and dest_addr not in set of internal registers.	Four 8-bit master byte writes (order is 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31]	1 2 1	If Write_Resp_En is not set, no write response. If Write_Resp_En is set, Write response on last CAZ Write response is concatenated if Write_Resp_Cat is set.	2
1,2	Quadlet write request to External CSR (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) and dest_addr not in set of internal registers.	None	4	None	2
0	Quadlet read request to External CSR (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) and dest_addr not in set of internal registers	Four 8-bit master byte reads (order is 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31]	2 1	Read response on last CAZ. Read response is concatenated if Read_Resp_Cat is set.	2
1,2	Quadlet read request to External CSR (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) and dest_addr not in set of internal registers.	None	4	None	2
0	Quadlet write request to Internal CSR (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) and dest_addr in set of internal registers	If CSR_WR_ECHO bit is set, four 8-bit master byte writes (order is 0,1,2,3) A[11:0] = dest_addr[20:31] + n D[7:0] = byte n of quadlet_data[0:31] If CSR_WR_ECHO bit is not set, no host transaction.	1 2 1	If Write_Resp_En is not set, no write response. If Write_Resp_En is set, Write response on last CAZ Write response is concatenated if Write_Resp_Cat is set.	3
1,2	Quadlet write request to Internal CSR (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) and dest_addr in set of internal registers.	None	4	None	3
0	Quadlet read request to Internal CSR (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) and dest_addr in set of internal registers.	None	2 1	Read response on last CAZ. Read response is concatenated if Read_Resp_Cat is set.	3
1,2	Quadlet read request to Internal CSR (Node_id, FFFFF,0000000 <= dest_addr <= Node_id, FFFFF,FFFFFFF) and dest_addr in set of internal registers.	None	4	None	3

Each of the four modes of operation is described in detail in the sections below.

Asynchronous Receive

In Microcontroller Emulation Mode, AADEC [1:0] = 0, controller 200 sources 8-bit Write and Read transactions on its host interface in response to Serial Bus Write or Read request transactions to CSR Micro_M_Reg. A Write request to Micro_M_Reg causes controller 200 to perform a Write transaction on its host interface with Address Am and Data Dm. The Least Significant Byte of the data written to Micro_M_Reg is asserted on D[7:0] and the next to least significant byte is asserted on A[11:0]. CSZ, CAZ, WEZ, AADEC, and HTYPE are part of the transaction with timing described in FIGURE 17. A Read request to Micro_M_Reg

causes controller 200 to do a Read transaction on its host interface. When the Read is acknowledged with CAZ, controller 200 will transmit a Read response. If CAZ occures within .9 microseconds of CSZ, the read response will be concatenated to the corresponding request acknowledge. Only one outstanding Read request is allowed. Timing for this operation is illustrated in FIGURE 18.

In Memory Mode, AADEC[1:0] = 1, controller 200 services Serial Bus quadlet or block Read or Write requests addressed to the node's Initial Memory Space using an 8-quadlet internal FIFO. Host interface to FIFO transfers are either 8-bit master or 16-bit slave.

During an 8-bit Master Write Request, a Write request (quadlet or block) to Initial Memory Space stores data in FIFO. Controller 200 does byte sized Write transactions using A, D, WEZ, CSZ, and CAZ with SRAM like timing shown in FIGURE 19. The address asserted on A[11:0] is the 12 LSBs of the destination address of the Write request.

During an 8-bit Master Read Request, controller 200 receives a Read request to Initial Memory Space and does byte sized transactions using A, D, WEZ, CSZ, and CAZ to Read data into the FIFO. When Read to FIFO is complete, controller 200 transmits a Read response. The timing for this operation is shown in FIGURE 20. The address asserted on A[11:0] is the 12 LSBs of the destination address of the Read request.

During a 16-bit Slave Write Request, a Write request (quadlet or block) to Initial Memory Space stores data in the FIFO 219. Data is then read from the FIFO 219 by an external agent using {A[7:0],D[7:0]} as a 16-bit data bus. REZ indicates that a Write request was received and that data is available in the FIFO. CSZ is used to indicate that the external master is ready to receive data. Each time data is clocked out of controller 200, an internal doublet FIFO pointer is incremented. When the internal doublet pointer equals requested data length, CAZ is used to indicate the last data and a Write response is transmitted. The timing for this operation is shown in FIGURE 21.

In a 16-bit Slave Read Request, when controller 200 receives a Read request to Initial Memory Space, response data must be written into the FIFO 219 using {A[7:0],D[7:0]} as a 16-bit data bus. WEZ indicates that a Read request was received and that space is available in the FIFO. CSZ is used to indicate that the external master is sourcing data. Each time data is clocked into controller 200, an internal doublet FIFO pointer is incremented. When the internal doublet pointer equals requested data length, CAZ is used to indicate the last data and a Read response is transmitted. The timing for this operation is shown in FIGURE 22.

In the External CSR mode, with AADEC[1:0] = 2, Write requests and Read requests to registers in Initial Register Space and Initial Unit Space not contained in controller 200 are serviced using a byte sized SRAM like transaction the same as that for the 8-bit Master Memory Mode.

In the Internal CSR mode, with AADEC[1:0] = 3, Write requests and Read requests to registers in Initial Register Space and Initial Unit Space that are contained in controller 200 are performed internally without activity on the host interface unless the CSR_WR_ECHO bit is set. In this case, a quadlet Write request results in the quadlet being echoed on the host interface using a byte sized SRAM like transaction the same as the 8-bit Master Memory Mode.

Asynchronous Transmit

Controller 200 can be programmed to transmit asynchronous Write requests automatically at regular intervals or manually. Only Write requests for data quadlet and data block formats are supported. In the automatic (or Sensor) mode, the host interface in 8-bit master mode reads 1 to 8 quadlets of packet data into the FIFO 219. This mode is store-and-forward and the amount of data read is determined by the data_length field in T_Control_Reg3. This Read starts at the Sensor_Prefetch_Event. Controller 200 then prepends an asynchronous header to the packet using information in T_Control_Reg(0:3) and then transmits the assembled asynchronous Write request at the Sensor_Transmit_Event. The Sensor_Prefetch_Event can be programmed to occur on either internal or external events using the Sensor_Prefetch_Event_Select field of T_Control_Reg0:

Sensor_Prefetch_Event_Select options:

1. Sensor prefetch disabled
2. Internal Cycle Start
3. Internal Cycle Done; and
4. External Rising Edge of TEVENT pin.

The Sensor_Transmit_Event can also be programmed to occur on similar events using the Sensor_Transmit_Event_Select field of T_Control_Reg0:

Sensor_Transmit_Event_Select options:

1. Sensor transmit disabled
2. Internal Cycle Start
3. Internal Cycle Done
4. External Falling Edge of TEVENT pin

In the manual asynchronous transmit mode, the host interface in 16-bit slave mode is used to load quadlet or block data into the FIFO 219. This mode is flow-through and transmit pipelining can occur. When the amount of data loaded into the FIFO reaches the Tx_Threshold or the Manual_Transmit_Event occurs, controller 200 will prepend an asynchronous header to the packet using information in AT_Control_Reg(0:3) and begin transmitting a Write request. The external host interface master must keep pace with the transmit rate or FIFO under run will occur. Flow control is provide to manage FIFO overrun. This is illustrated in the timing diagram in FIGURE 21.

The Manual_Transmit_Event can be programmed to occur using the Manual_Transmit_Event_Select field of T_Control_Reg0:

Manual_Transmit_Event_Select options:

1. Internal Cycle Start
2. Internal Cycle Done
3. Manual_Transmit_Event bit set to one

4. External Falling Edge of TEVENT pin

The four transmit control registers have a plurality of fields, the fields being illustrated in FIGURE 14. The definition of these fields is set forth in the following *Table 3*.

Table 3: T_Control_Reg(0:4) Field Definitions

Field Mnemonic	Field Width	Field Description
STES	2	Sensor Transmit Event Select
SPES	2	Sensor Prefetch Event Select
MTES	2	Manual Transmit Event Select
MTE	1	Manual Transmit Event
tl	6	transaction label
rt	2	retry code
tcode	4	transaction code
pri	4	priority
destination_ID	16	address
destination_offset_lo	16	address
destination_offset_hi	32	address
data_length	16	data length used for asynchronous and isochronous transmits
DMEN	2	Data Mover Enable 00 = Automatic asynchronous mode 01 = Manual asynchronous mode 10 = Automatic isochronous mode 11 = Manual isochronous mode
Tx_Threshold	2	Transmit threshold 00 = 2 quadlets (8 bytes) 01 = 4 quadlets (16 bytes) 10 = 6 quadlets (24 bytes) 11 = 8 quadlets (32 bytes)
tag	2	iso header tag
channel	6	iso header channel number
sy	4	iso header syncronization bits

Isochronous Transmit

Controller 200 can be programmed to transmit isochronous data block packets in much the same way as asynchronous transmits, described hereinabove. When the DMEN field of T_Control_Reg4 is set to Automatic or Manual Isochronous mode, controller 200 will function identically to asynchronous transmit mode except that an isochronous header will be prepended to the packet using information in T_Control_Reg3 and T_Control_Reg4.

Host CSR Access

Asserting {10} on HTYPE[1:0] puts controller 200's host interface in CSR access mode. In this mode, the host interface functions as a slave. Internal CSR registers can be written to or read from. Address A[11:0] is used to address the internal 4K CSR space. Cycle Start (CAZ) is an input used to start the transaction. Write (WEZ) is an input used to indicate Write or Read Cycle acknowledge. (CAZ) is asserted by controller 200 to end the transaction. The timing for this operation is illustrated in FIGURE 15.

Referring now to FIGURE 16, there is illustrated a diagrammatic view of the controller 200 illustrating the various address, data and control pins on the host interface side and the various interfaces with the physical link layer 54. It can be seen that there is provided a 12-bit address bus for the bus 204 and an 8-bit data bus for the bus 206. *Table 4* illustrates the various terminal functions.

Table 4: Terminal Functions

Signal Name	Pin #	I/O	Description
A[11:0]		I/O	Address in 8-bit Master and CSR access Host Interface mode.
D[7:0]		I/O	Data in all Host Interface modes.
CSZ		I/O	Output Cycle Start/Chip Select in 8-bit Master mode. Input Cycle Start in CSR access mode
CAZ		I/O	Input Cycle Acknowledge for 8-bit Master transactions. Input Write or Read strobe for 16-bit Slave transactions. Output Cycle Acknowledge for CSR access mode
WEZ		I/O	Output Write indication for 8-bit Master transactions. Input Write indication for CSR access mode
REZ			
HCLK		O	Host Clock, Programmable using HCLK_SEL 00 = SYSCLK 01 = SYSCLK/2 10 = SYSCLK/4 11 = SYSCLK/8
AADEC[1:0]		O	Asynchronous Address Decode 00 = Microcontroller Emulation, Decoded CSR address Micro_M_Reg. 01 = Memory, Decoded Initial Memory Space 10 = External CSR, Decoded Register Space not in controller 200 11 = Internal CSR, Decoded Register Space in controller 200
HTYPE[1:0]		I	Host Interface Type 00 = 8-bit Master controller 200 asserts A (address), CSZ, and WR. D (data) is bidirectional depending on WRZ. External agent uses CAZ to acknowledge transaction. 01 = 16-bit Slave

			Implied address Auto Sequence 10 = CSR access mode In this mode, the host interface functions as a slave. Read and Write transactions can be accomplished using A, D, CSZ, CAZ, and WRZ to access the 4K CSR space.
TEVENT		I	Rising edge triggers Sensor_Prefetch_Event Falling edge triggers Sensor_Transmit_Event or Manual_Transmit_Event.
D[0:1]		I/O	Data 0-1 of the phy-link data bus. Data is expected on D[0:1] at 100 Mb/s. D[0] is MS bit.
CTL[0:1]		I/O	Control 0-1 of the phy-link control bus. CTL[0] is MS bit.
LREQ		O	Link request to Phy. This output is used to make bus requests and to access the Phy registers.
SYSCLK		I	System Clock. This input is a 49.152 Mhz from the Phy.

Referring now to FIGURE 17, there is illustrated a timing diagram for the operation wherein a quadlet Write request was received in the asynchronous mode for interfacing with the host. A Write request is received, as illustrated by block 300. After a short propagation from the rising edge of the clock HCLK, CSZ falls low, indicating the output cycle start/chip select operation. Both the data and the address are decoded and placed onto the address and data buses, and the WEZ line also goes low with the CSZ line, indicating an output Write operation. The AADEC line is set to "0" during this period. While CSZ and WEZ are low, an acknowledgment is sent, followed by a Write response.

Referring now to FIGURE 18, there is illustrated a timing diagram wherein the quadlet Read request is serviced. The CSZ line will go low after a short delay, followed by the CAZ going low, the CAZ indicating an input cycle acknowledge signal for the input-master transaction. This will be acknowledged to the master, and the address for the Read operation will be placed onto the bus when CSZ goes low, and then data will be placed onto the bus from the peripheral device from the CAZ going low. This data will then be stored in the FIFO 219 and sent to the requesting end as the response.

Referring now to FIGURE 19, there is illustrated a timing diagram for the operation wherein a quadlet Write request has been received in the memory mode. These are block Read or Write requests addressed to the node's initial memory space using the internal FIFO 219. This is similar to SRAM timing in that the block of data

will be sequentially processed going low, rising high, and then falling low for three more cycles. This will result in four addresses and associated data being received and written.

Referring now to FIGURE 20, there is illustrated a timing diagram depicting a Read request in the memory mode. This operates similar to that described hereinabove with reference to FIGURE 19, with the exception that it will be repeated four times, since this is a Read request to initial memory space with byte sized transactions being performed. The appropriate Read response is sent after the FIFO 219 has been filled.

In summary, there has been provided a link/transaction layer controller that incorporates therein a microcontroller emulator. This microcontroller emulator allows the link/transaction layer to incorporate the operation of a microcontroller therein such that a transmitting node can virtually address the address space of the peripheral device to transmit data thereto during a Write operation and read data therefrom during a Read operation without having to go through a separate microcontroller and the necessary interface transactions. Therefore, the remote transmitting node can directly control the location that is accessed and essentially perform some of the microcontroller operations at the requesting node.

Although the preferred embodiment has been described in detail, it should be understood that various changes, substitutions and alterations can be made therein without departing from the spirit and scope of the invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A serial bus interface disposed on a local node for interfacing between a serial bus and a host system and for receiving information from the serial bus placed thereon by a remote node and transferring this received data to the host system, and receiving information from the host system and transferring the received information to the serial bus for reception by the remote node, comprising:

a data receiver for receiving data generated by the remote node from the serial bus;

a data transmitter for transmitting data to the serial bus for receipt at the remote node;

a plurality of registers, at least one of said registers addressable by the remote node for storage of received data therein;

said data receiver operable to store received data in said at least one register during a Read operation;

said data transmitter for transmitting data to said serial bus during a Write operation; and

a host bus interface for interfacing directly with a host bus on the host system, said host bus interface for transferring data stored in said at least one register to the host bus during a Write operation when the data is received and stored in said at least one register and retrieving data from the host bus during a Read operation.

2. The serial bus interface of Claim 1, wherein said serial bus interface includes a standard register space, wherein said at least one register occupies a portion of said standard register space.

3. The serial bus interface of Claim 1, wherein select ones of said plurality of registers are dedicated to storage of standard bus interface information and wherein a remote node can directly address said plurality of registers associated with said standard bus interface information for access of said information therefrom and wherein said data receiver is operable to recognize a request for access to one of said

plurality of registers and said data transmitters are operable to transmit the contents thereof when addressed.

4. The serial bus interface of Claim 3, wherein select ones of said plurality of registers comprise configuration registers which said configuration registers are utilized for configuration information that define the operation of the serial bus interface such that a remote node can program the operation of the serial bus interface by accessing one of said configuration registers.

5. The serial bus interface of Claim 1, wherein data received by said data receiver comprises data packets and data transmitted by said data transmitter comprises data packets, said data packets comprising information necessary to identify the transmitting node on the serial bus and the content of said data packet and information identifying the remote node designated to receive the data packet.

6. The serial bus interface of Claim 5, wherein each data reception or data transmission operation is preceded by a data request, a Write request or a Read request, respectively, from the remote node, which requests are contained in said received data packet, said received data packet associated with a Write request containing the data associated therewith, which data is stored in said at least one register and said host interface recognizing the Write request and transferring said data stored in said at least one register to the host bus, and in a Read request, said host interface recognizing the said Read request and accessing the data from the host system for transfer to the remote node with said data transmitter.

7. The serial bus interface of Claim 6, wherein said Write request from the remote node includes both address and data information for storage in said at least one register and wherein said host interface is operable to transmit both the address information and the data information to the host bus, the host bus having an address bus and a data bus.

8. The serial bus interface of Claim 6, wherein said Read request from the remote node includes an address which is stored in said at least one register and said

host interface is operable to transmit said address to the host bus upon recognizing a Read request, the host bus having an address bus and a data bus, and retrieving data from the data bus for transmission to the remote node by said data transmitter.

9. A method for interfacing between a serial bus and a host system on a local node and for receiving information from the serial bus placed thereon by a remote node and transferring this received data to the host system, and receiving information from the host system and transferring the received information to the serial bus for reception by the remote node, comprising the steps of:

receiving with a data receiver data generated by the remote node from the serial bus;

transmitting data to the serial bus for receipt at the remote node;

providing a plurality of registers, at least one of the registers addressable by the remote node for storage of received data therein;

said data receiver operable to store received data in said at least one register during a Read operation;

said data transmitter for transmitting data to said serial bus during a Write operation; and

a host bus interface for interfacing directly with a host bus on the host system, said host bus interface for transferring data stored in the at least one register to the host bus during a Write operation when the data is received and stored in the at least one register and retrieving data from the host bus during a Read operation.

10. The method of Claim 9, wherein the serial bus interface includes a standard register space, wherein the at least one register occupies a portion of the standard register space.

11. The method of Claim 9, wherein select ones of the plurality of registers are dedicated to storage of standard bus interface information and further comprising directly addressing by a remote node the plurality of registers associated with the standard bus interface information for access of the information therefrom and wherein the step of receiving is operable to recognize a request for access to one of the plurality of registers and the step of transmitting is operable to transmit the contents thereof when addressed.

12. The method of Claim 11, wherein select ones of the plurality of registers comprise configuration registers which configuration registers are utilized for configuration information that define the operation of the serial bus interface such that a remote node can program the operation of the serial bus interface by accessing one of the configuration registers.

13. The method of Claim 9, wherein data received by the step of receiving comprises data packets and data transmitted by the step of transmitting comprises data packets, the data packets comprising information necessary to identify the transmitting node on the serial bus and the content of the data packet and information identifying the remote node designated to receive the data packet.

14. The method of Claim 13, wherein each data reception or data transmission operation is preceded by a data request, a Write request or a Read request, respectively, from the remote node, which requests are contained in the received data packet, the received data packet associated with a Write request containing the data associated therewith, which data is stored in the at least one register and the host interface recognizing the Write request and transferring the data stored in the at least one register to the host bus, and in a Read request, the host interface recognizing the Read request and accessing the data from the host system for transfer to the remote node in the step of transmitting.

15. The method of Claim 14, wherein the Write request from the remote node includes both address and data information for storage in the at least one register and wherein the host interface is operable to transmit both the address information and the data information to the host bus, the host bus having an address bus and a data bus.

16. The method of Claim 14, wherein the Read request from the remote node includes an address which is stored in the at least one register and the host interface is operable to transmit the address to the host bus upon recognizing a Read request, the host bus having an address bus and a data bus, and retrieving data from the data bus for transmission to the remote node by the data transmitter.

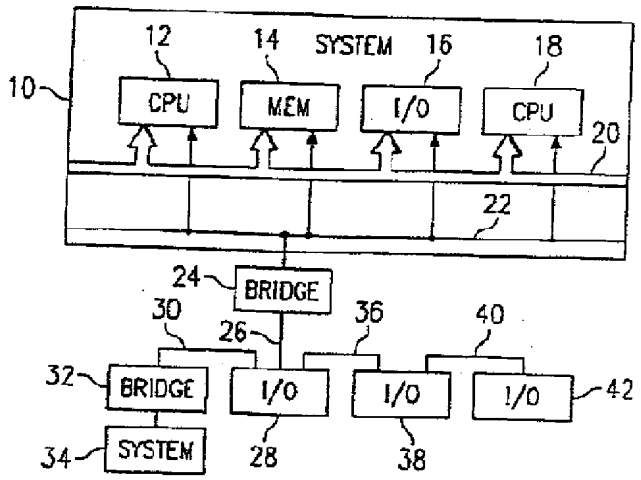


FIG. 1

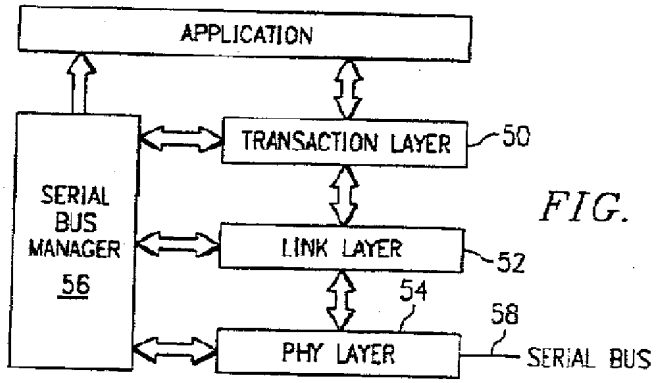


FIG. 2

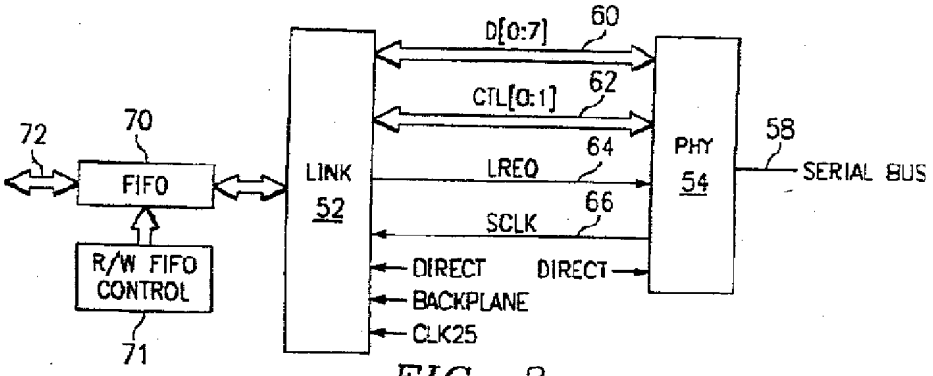


FIG. 3

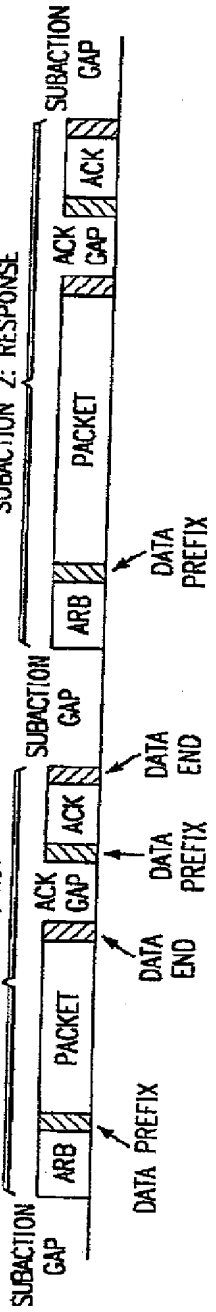

SUBACTION 1: REQUEST
SUBACTION 2: RESPONSE
SUBACTION GAP
ARB
PACKET
ACK GAP
ACK
DATA PREFIX
DATA END
DATA PREFIX
DATA END
SUBACTION GAP
ARB
PACKET
ACK GAP
ACK
SUBACTION GAP
DATA PREFIX

FIG. 4

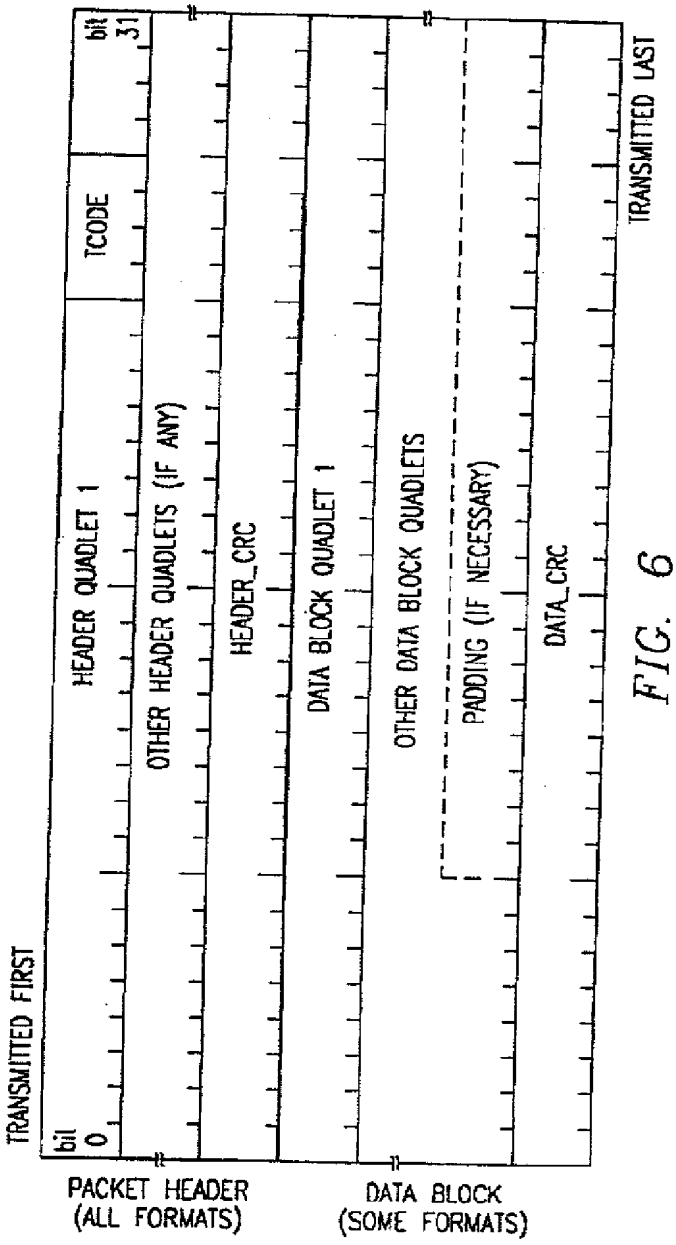

TRANSMITTED FIRST
bit 0
HEADER QUADLET 1
TCODE
bit 31
OTHER HEADER QUADLETS (IF ANY)
HEADER_CRC
DATA BLOCK QUADLET 1
OTHER DATA BLOCK QUADLETS
PADDING (IF NECESSARY)
DATA_CRC
TRANSMITTED LAST
PACKET HEADER (ALL FORMATS)
DATA BLOCK (SOME FORMATS)

FIG. 6

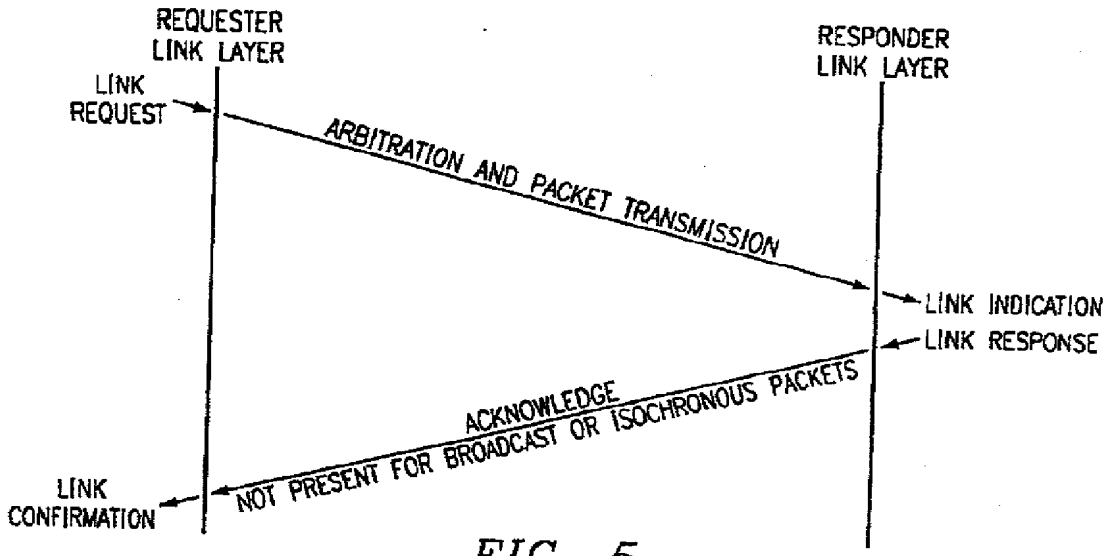

REQUESTER
LINK LAYER
RESPONDER
LINK LAYER
LINK
REQUEST
ARBITRATION AND PACKET TRANSMISSION
LINK INDICATION
LINK RESPONSE
ACKNOWLEDGE
NOT PRESENT FOR BROADCAST OR ISOCHRONOUS PACKETS
LINK
CONFIRMATION

FIG. 5

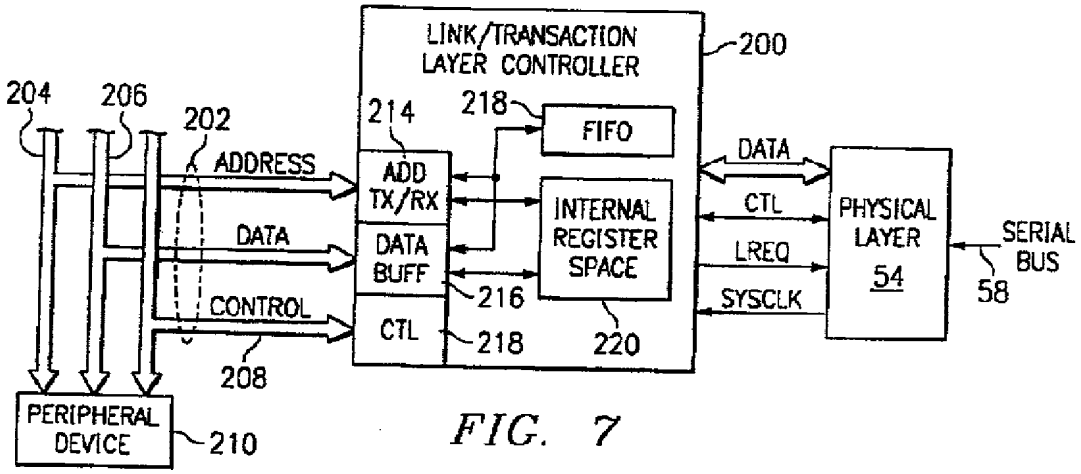

LINK/TRANSACTION
LAYER CONTROLLER
200
204
206
202
214
218
FIFO
ADDRESS
ADD
TX/RX
DATA
DATA
BUFF
CONTROL
CTL
216
218
220
INTERNAL
REGISTER
SPACE
208
DATA
CTL
LREQ
SYSCLK
PHYSICAL
LAYER
54
SERIAL
BUS
58
PERIPHERAL
DEVICE
210

FIG. 7

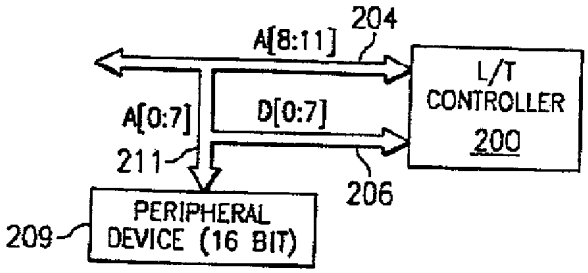


FIG. 7a

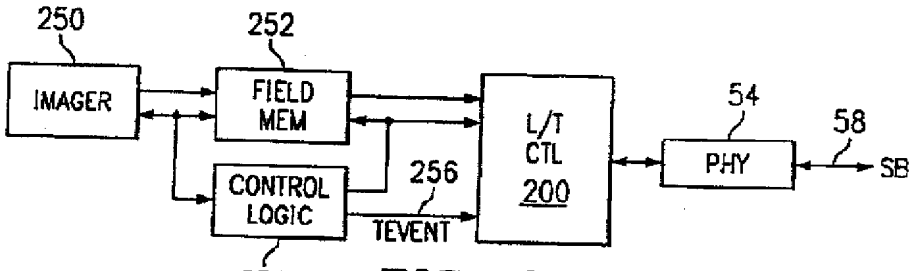


FIG. 8

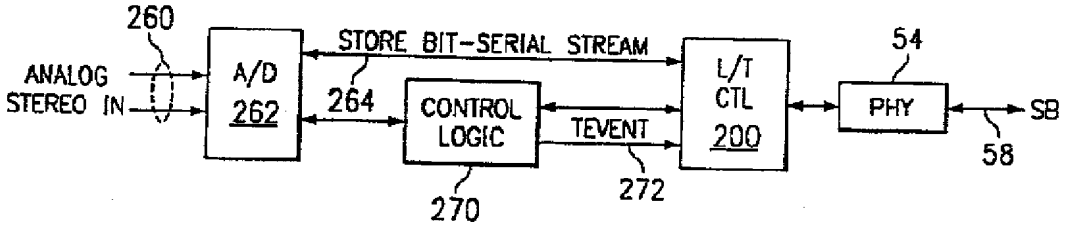
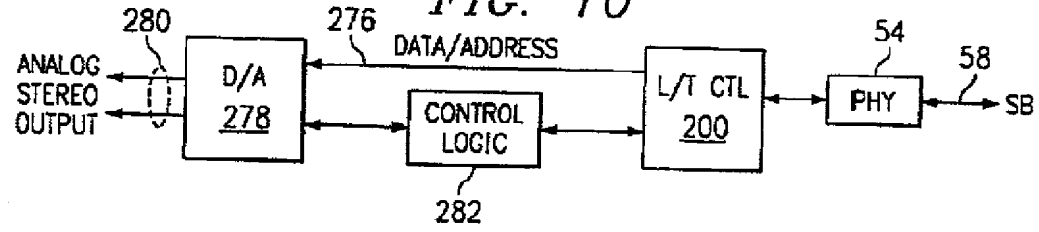


FIG. 9

FIG. 10



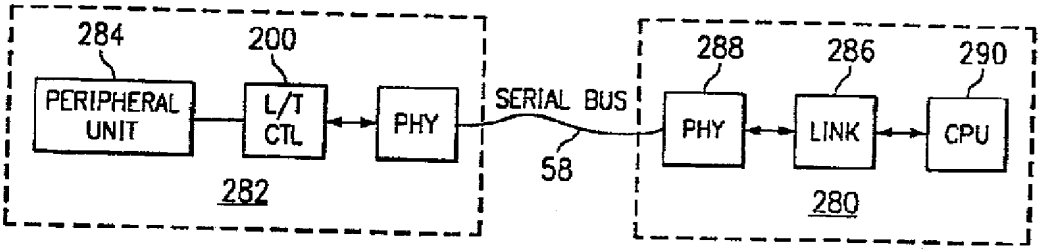


FIG. 11

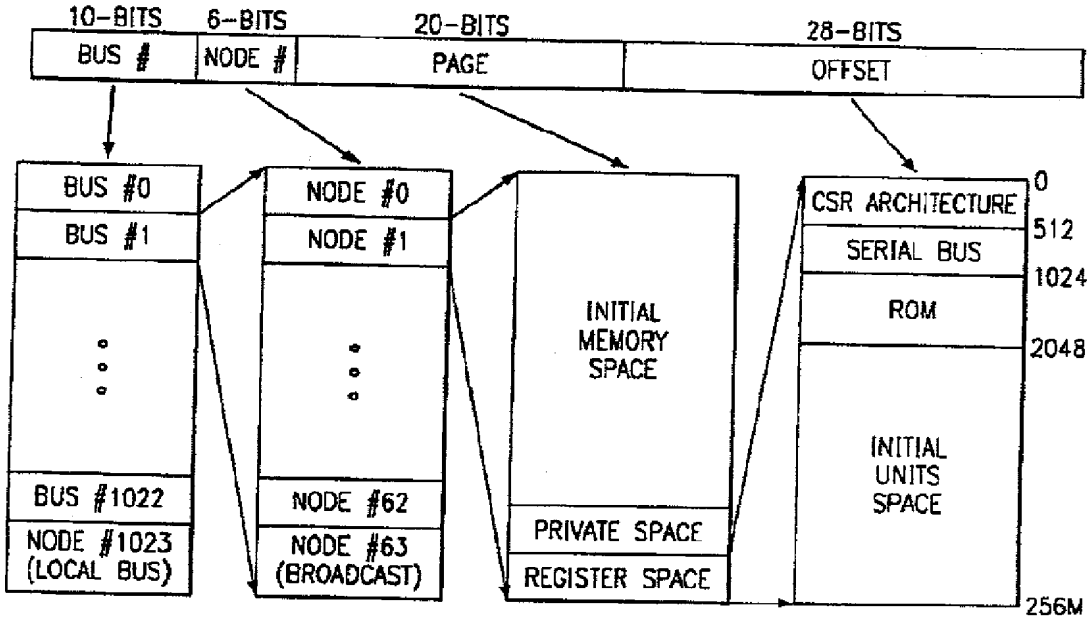


FIG. 12

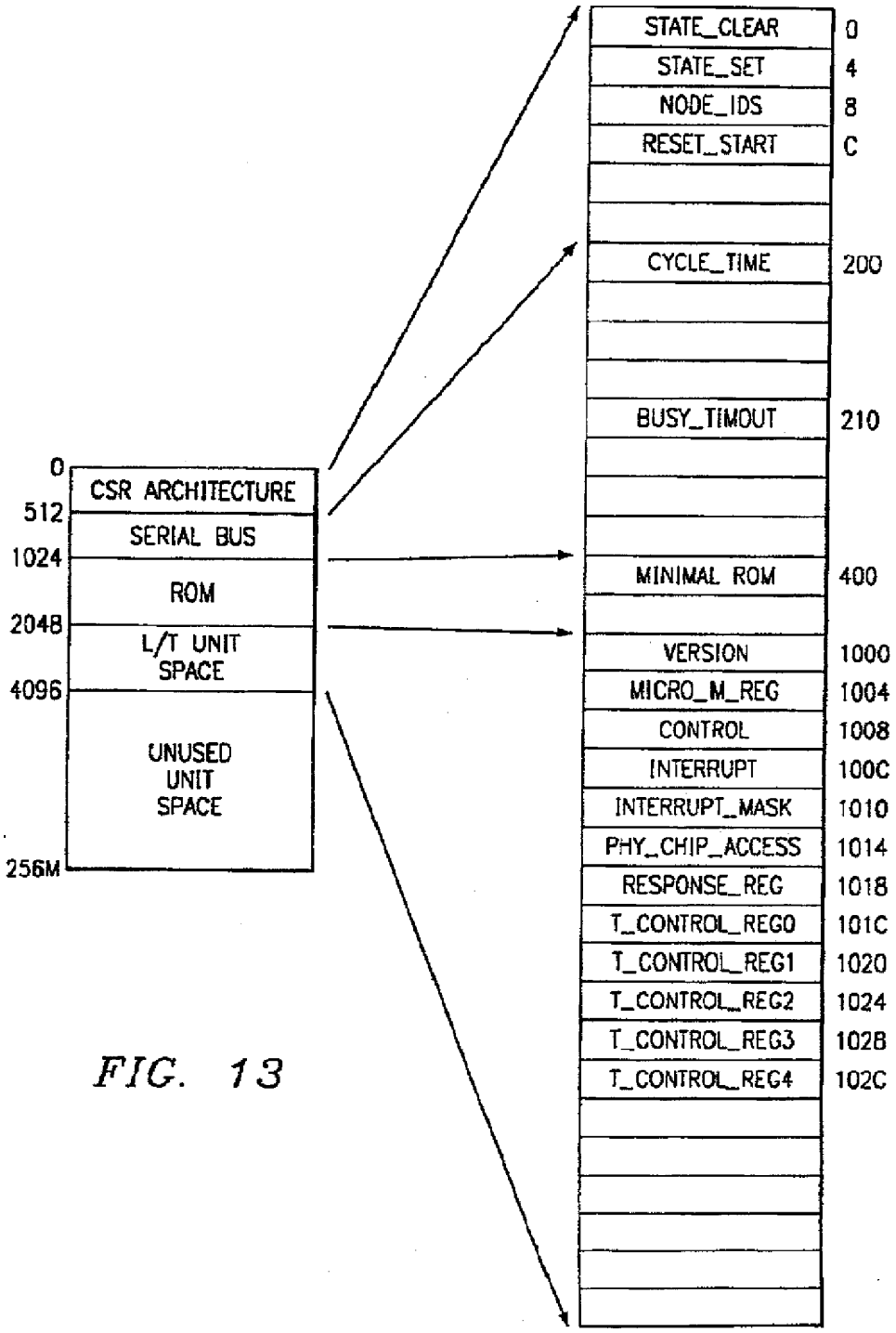

0
CSR ARCHITECTURE
512
SERIAL BUS
1024
ROM
2048
L/T UNIT SPACE
4096
UNUSED UNIT SPACE
256M
STATE_CLEAR 0
STATE_SET 4
NODE_IDS 8
RESET_START C
CYCLE_TIME 200
BUSY_TIMOUT 210
MINIMAL ROM 400
VERSION 1000
MICRO_M_REG 1004
CONTROL 1008
INTERRUPT 100C
INTERRUPT_MASK 1010
PHY_CHIP_ACCESS 1014
RESPONSE_REG 1018
T_CONTROL_REG0 101C
T_CONTROL_REG1 1020
T_CONTROL_REG2 1024
T_CONTROL_REG3 1028
T_CONTROL_REG4 102C

FIG. 13

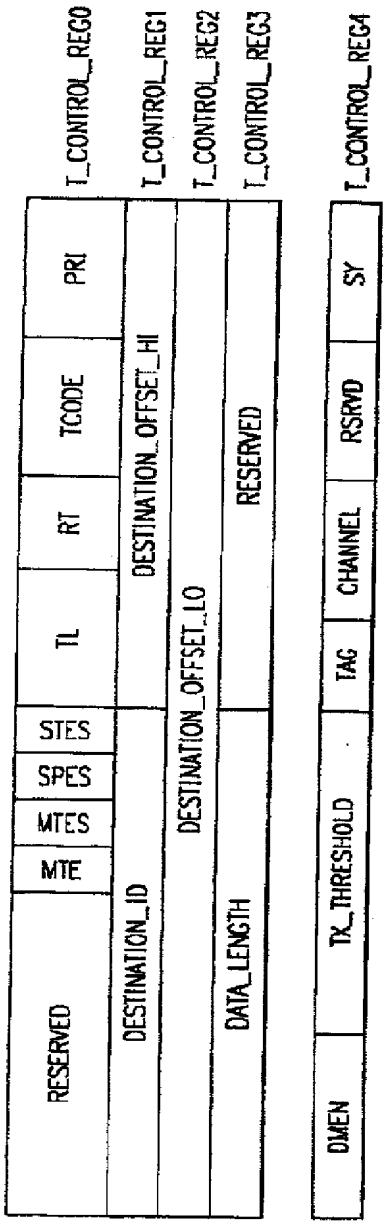


FIG. 14

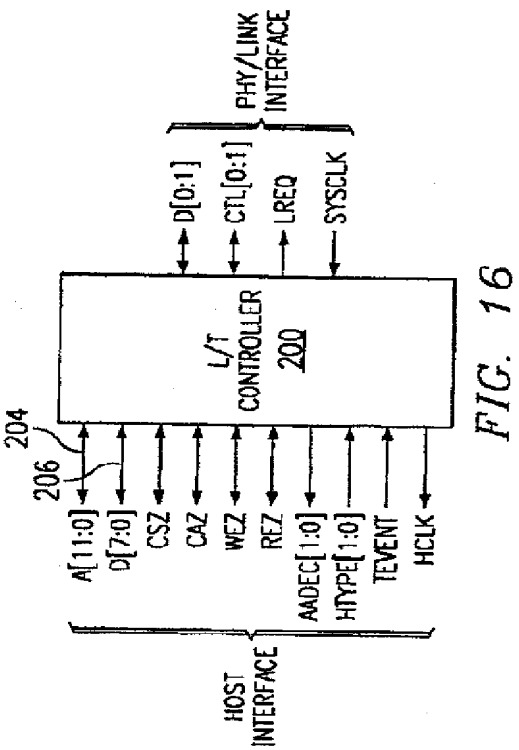


FIG. 16

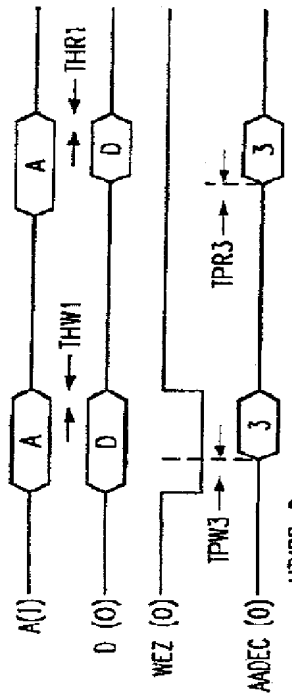


FIG. 15

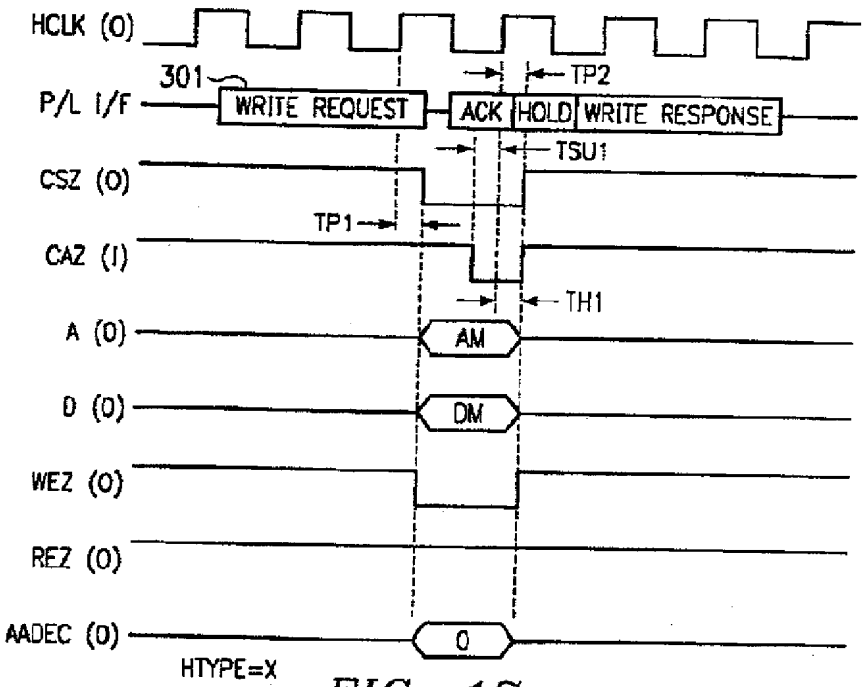

HCLK (O)
301
P/L I/F
WRITE REQUEST
TP2
ACK
HOLD
WRITE RESPONSE
TSU1
CSZ (O)
TP1
CAZ (I)
TH1
A (O)
AM
D (O)
DM
WEZ (O)
REZ (O)
AADEC (O)
O
HTYPE=X

FIG. 17

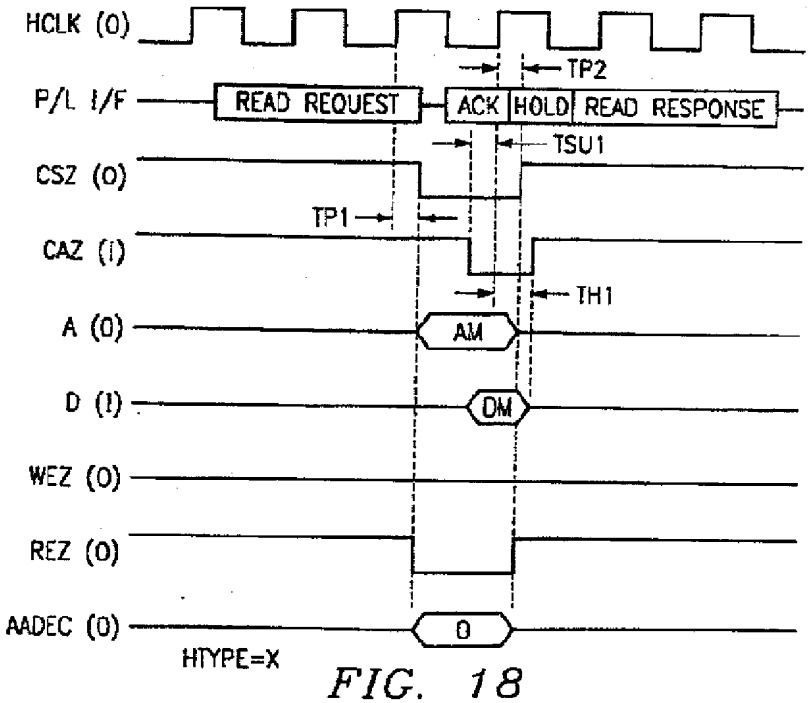

HCLK (O)
TP2
P/L I/F
READ REQUEST
ACK
HOLD
READ RESPONSE
TSU1
CSZ (O)
TP1
CAZ (I)
TH1
A (O)
AM
D (I)
DM
WEZ (O)
REZ (O)
AADEC (O)
O
HTYPE=X

FIG. 18

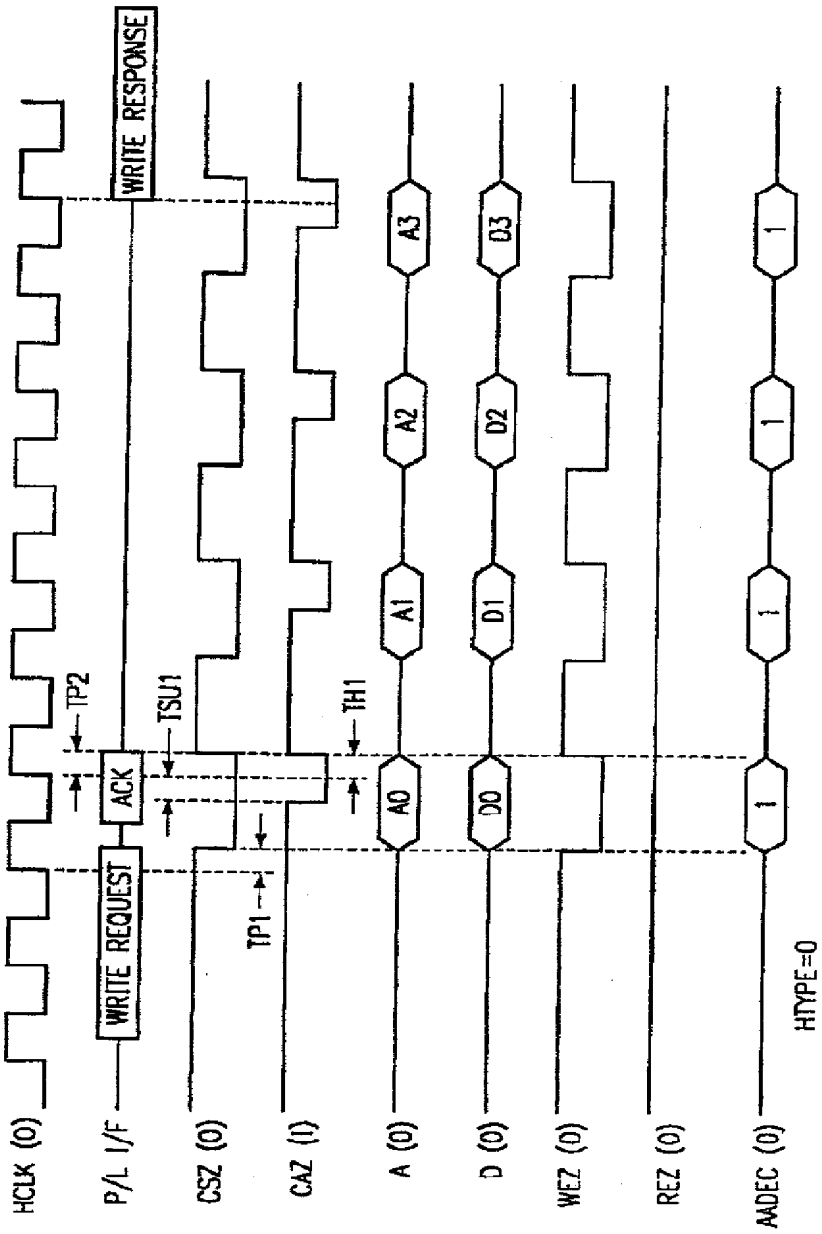

HCLK (0)
P/L I/F
CSZ (0)
CAZ (1)
A (0)
D (0)
WEZ (0)
REZ (0)
AADEC (0)
WRITE REQUEST
ACK
WRITE RESPONSE
TP2
TSU1
TP1
TH1
A0
A1
A2
A3
D0
D1
D2
D3
1
HTYPE=0

FIG. 19

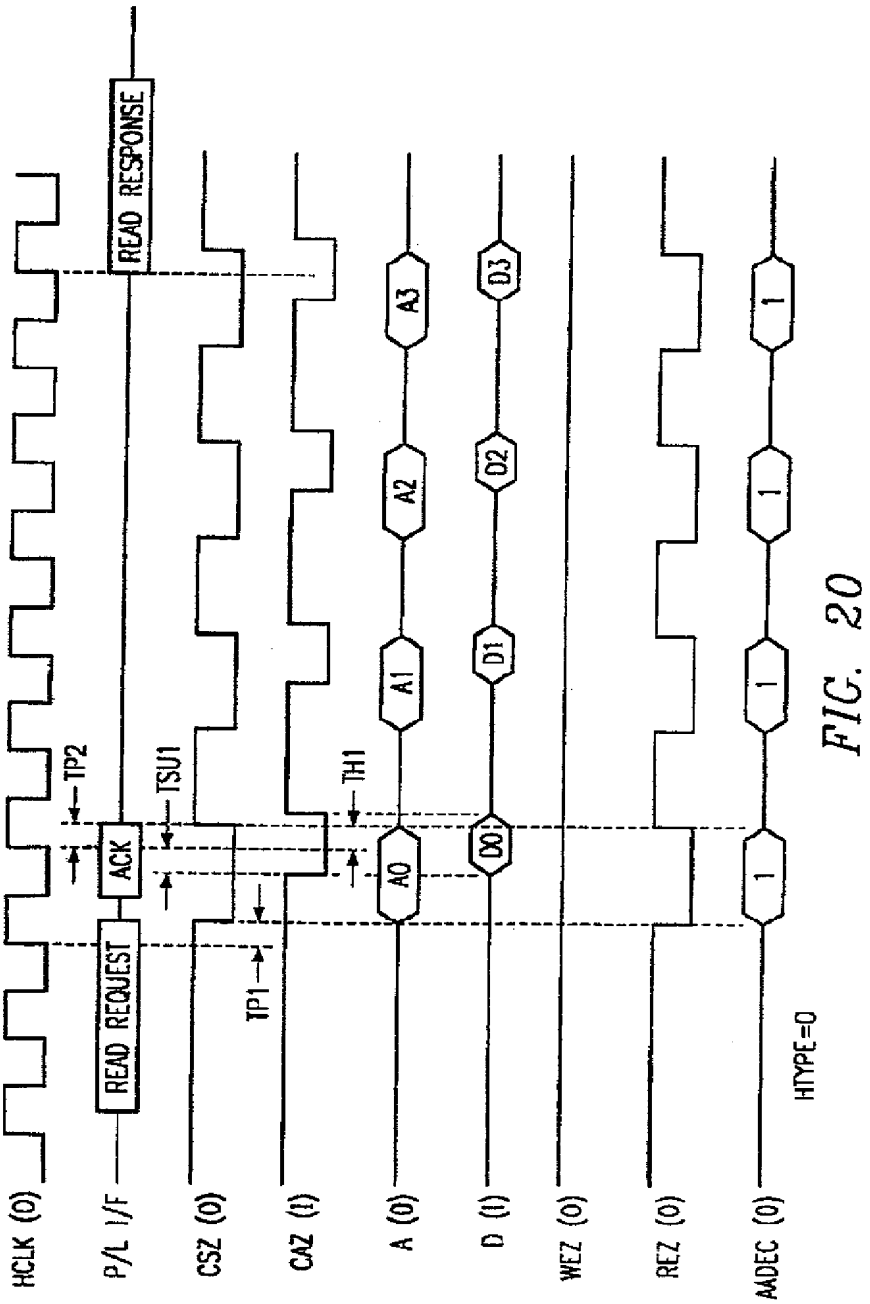

HCLK (0)
P/L I/F
CSZ (0)
CAZ (1)
A (0)
D (1)
WEZ (0)
REZ (0)
AADEC (0)
READ REQUEST
ACK
READ RESPONSE
TP2
TSU1
TP1
TH1
A0
A1
A2
A3
D0
D1
D2
D3
1
HTYPE=0

FIG. 20

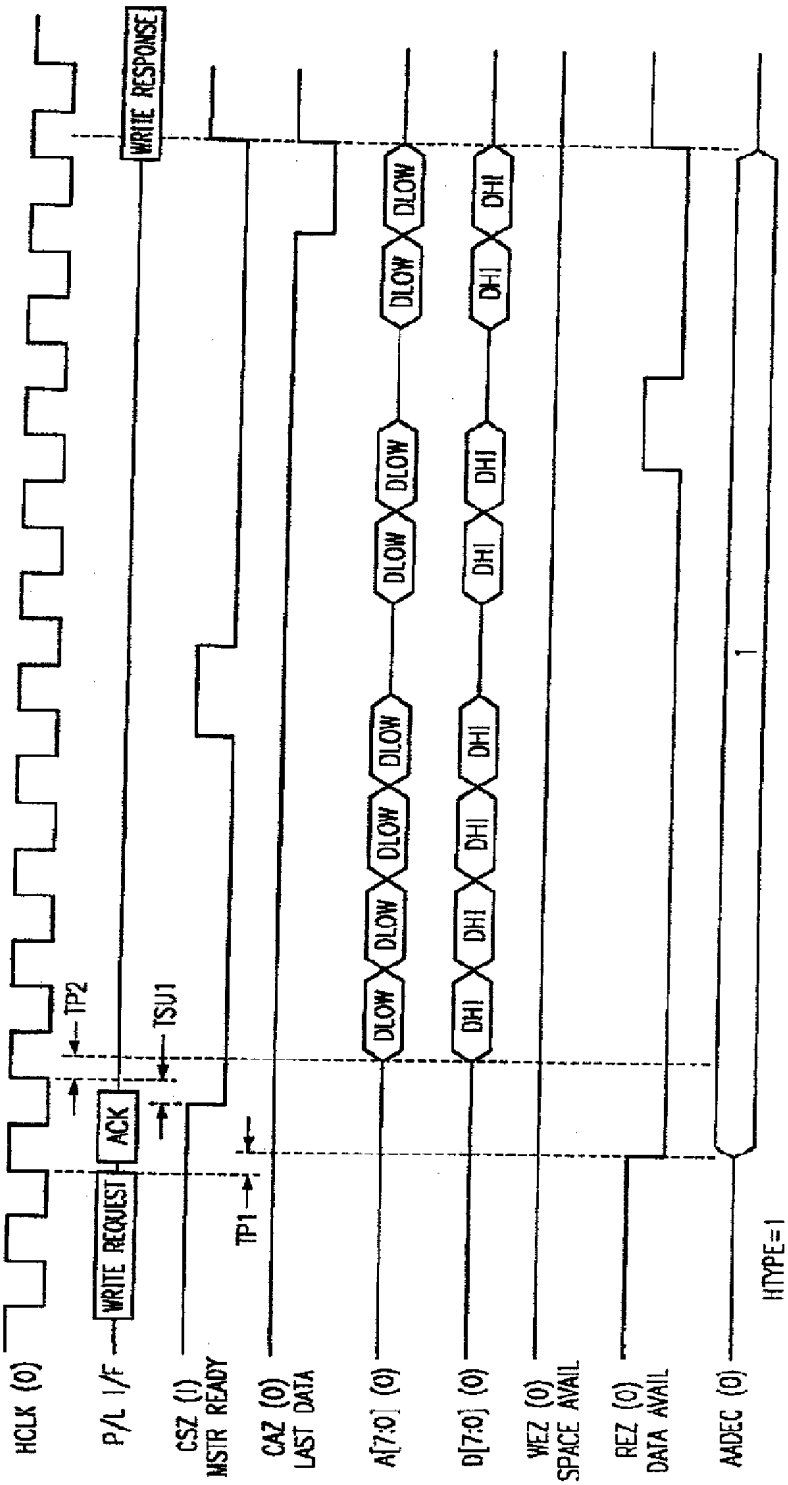

HCLK (0)
P/L I/F
CSZ (1)
MSTR READY
CAZ (0)
LAST DATA
A[7:0] (0)
D[7:0] (0)
WEZ (0)
SPACE AVAIL
REZ (0)
DATA AVAIL
AADEC (0)
WRITE REQUEST
ACK
WRITE RESPONSE
TP2
TSU1
TP1
DLOW
DHI
1
HTYPE=1

FIG. 21

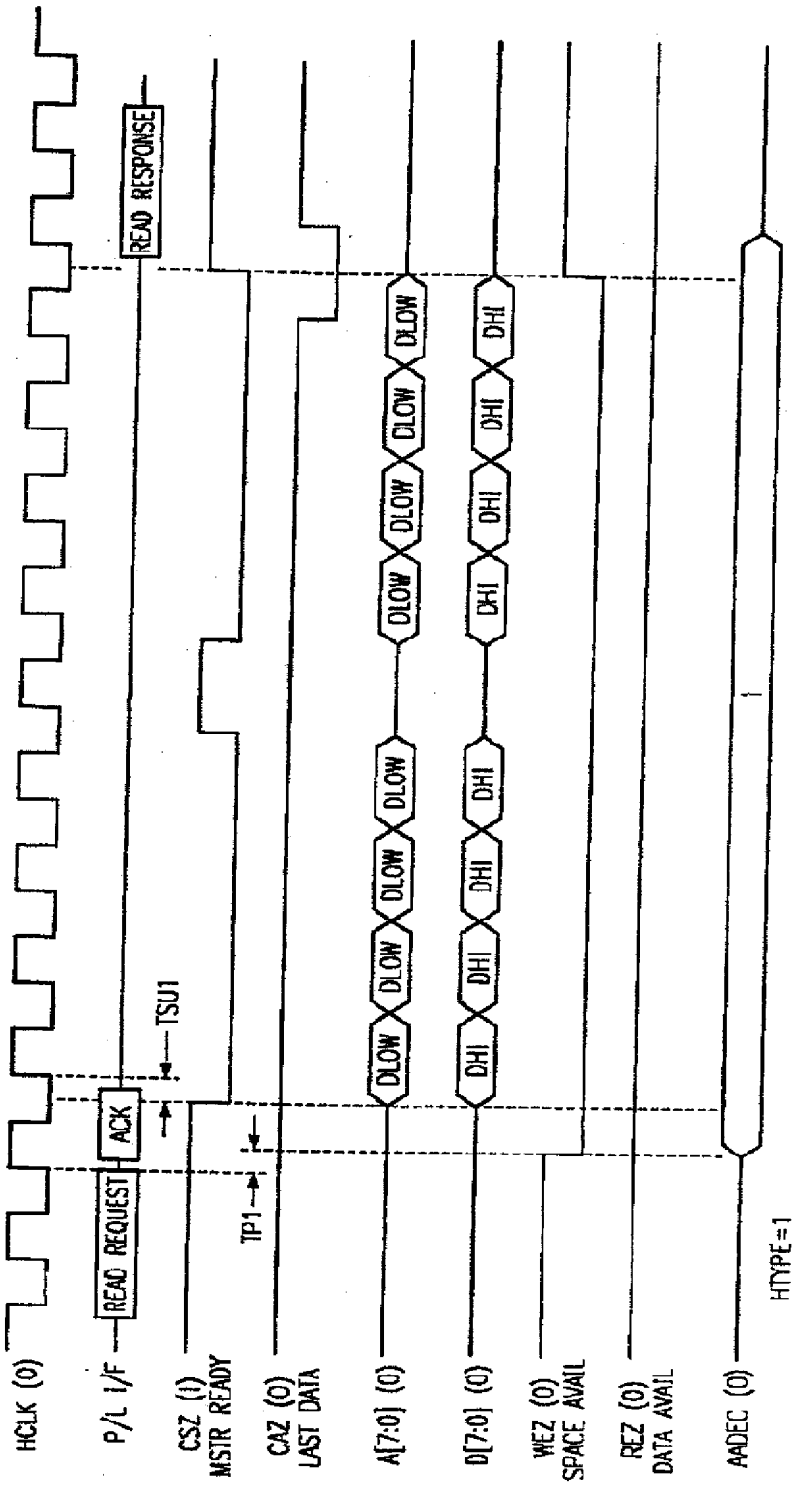

HCLK (0)
P/L I/F
CSZ (1)
MSTR READY
CAZ (0)
LAST DATA
A[7:0] (0)
D[7:0] (0)
WEZ (0)
SPACE AVAIL
REZ (0)
DATA AVAIL
AADEC (0)
READ REQUEST
ACK
READ RESPONSE
TSU1
TP1
DLOW
DHI
HTYPE=1

FIG. 22

LINK/TRANSACTION LAYER CONTROLLER WITH INTEGRAL MICROCONTROLLER EMULATION

1 Abstract

ABSTRACT OF THE DISCLOSURE

An IEEE 1394 serial bus (58) is interfaced utilizing a physical layer (54) to extract the data and a link/transaction layer controller (200) to interface the data from the physical layer (54) to a host system. The host system consists of a peripheral device (210) which is interfaced with various host buses (202). The link/transaction layer controller (200) is operable to emulate the microcontroller function such that addresses can be transmitted to the peripheral device (210) along with data, such that a remote node can access the peripheral device (210) by transmitting address and data information thereto. Alternatively, address and data information can be transmitted from the peripheral device (210) to the controller (200) which will process the received address and data and transmit it to a remote node.

2 Representative Drawing

Fig 7

【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第７部門第３区分
【発行日】平成18年11月30日(2006.11.30)

【公開番号】特開2000-188626(P2000-188626A)
【公開日】平成12年7月4日(2000.7.4)
【出願番号】特願平11-327311
【国際特許分類】
　Ｈ０４Ｌ　29/10　(2006.01)
　Ｇ０６Ｆ　13/38　(2006.01)
　Ｇ０６Ｆ　13/42　(2006.01)
　Ｈ０４Ｌ　12/40　(2006.01)
【ＦＩ】
　Ｈ０４Ｌ　13/00　３０９Ｃ
　Ｇ０６Ｆ　13/38　３５０
　Ｇ０６Ｆ　13/42　３２０Ｃ
　Ｈ０４Ｌ　12/40　Ｚ

【手続補正書】
【提出日】平成18年10月13日(2006.10.13)
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正の内容】
【特許請求の範囲】
　【請求項１】　シリアル・バスとホスト・システム間をインタフェースするため、および、遠隔ノードにより前記シリアル・バス上に置かれた情報を前記シリアル・バスから受信しかつ該受信した情報を前記ホスト・システムに転送し、また、前記ホスト・システムから情報を受信しかつ該受信した情報を前記遠隔ノードによる受信のため前記シリアル・バスに転送するためにローカル・ノード上に設けられたシリアル・バス・インタフェースであって、
　前記遠隔ノードにより生成されたデータを前記シリアル・バスから受信するためのデータ受信器と、
　データを前記遠隔ノードでの受信のため前記シリアル・バスに送信するためのデータ送信器と、
　少なくとも１つのレジスタがそこに受信データを記憶のため前記遠隔ノードによりアドレス可能である複数のレジスタを備えていて、
　前記データ受信器は読み出し動作の間受信したデータを前記少なくとも１つのレジスタに記憶するよう動作可能であり、
　また、前記データ送信器は書き込み動作の間データを前記シリアル・バスに送信するようになっており、
　さらに、前記ホスト・システム上のホスト・バスに直接インタフェースするためのホスト・バス・インタフェースを備えていて、該ホスト・バス・インタフェースは、前記少なくとも１つのレジスタに記憶されたデータを、書き込み動作の間、該データが受信されて前記少なくとも１つのレジスタに記憶されるとき、前記ホスト・バスに転送し、また、読み出し動作の間データを前記ホスト・バスから取り出すようになっているシリアル・バス・インタフェース。
　【請求項２】　請求項１記載のシリアル・バス・インタフェースにおいて、シリアル・バス・インタフェースが標準レジスタ空間を含んでいて、前記少なくとも１つのレジス

タは該標準レジスタ空間の一部を占有しているシリアル・バス・インタフェース。

【請求項３】　請求項１記載のシリアル・バス・インタフェースにおいて、前記複数のレジスタのうちの選ばれたレジスタは標準バス・インタフェース情報の記憶に専用に供されていて、遠隔ノードは前記標準バス・インタフェース情報に関連付けられた前記複数のレジスタを直接アドレスしてそこから該情報にアクセスでき、また、前記データ受信器は前記複数のレジスタのうちの１つへのアクセス要求を認識するよう動作可能でかつ前記データ送信器はアドレスされるときその内容を送信するよう動作可能であるシリアル・バス・インタフェース。

【請求項４】　請求項３記載のシリアル・バス・インタフェースにおいて、前記複数のレジスタのうちの選ばれたレジスタはコンフィギュレーション・レジスタから成っていて、該コンフィギュレーション・レジスタは前記シリアル・バス・インタフェースの動作を定めるコンフィギュレーション情報のために用いられ、それにより、遠隔ノードは前記コンフィギュレーション・レジスタの１つにアクセスすることによって前記シリアル・バス・インタフェースの動作をプログラムできるようになっているシリアル・バス・インタフェース。

【請求項５】　請求項１記載のシリアル・バス・インタフェースにおいて、前記データ受信器によって受信されるデータおよび前記データ送信器によって送信されるデータはデータ・パケットであり、該データ・パケットは前記シリアル・バス上の送信ノードを識別するのに必要な情報と、該データ・パケットの内容と、データ・パケットを受信するよう指定された遠隔ノードを識別する情報を含んでいるシリアル・バス・インタフェース。

【請求項６】　請求項５記載のシリアル・バス・インタフェースにおいて、各データ受信または送信動作に先行して前記遠隔ノードからの書き込み要求または読み出し要求のデータ要求があり、該要求は前記受信されたデータ・パケット内に含まれており、書き込み要求に関連した前記受信データ・パケットは前記少なくとも１つのレジスタに記憶されたそれに関連したデータを含んでいて、前記ホスト・インタフェースはこの書き込み要求を認識して前記少なくとも１つのレジスタに記憶された前記データを前記ホスト・バスに転送し、読み出し要求では、前記ホスト・インタフェースはこの読み出し要求を認識して該データに前記データ送信器を持つ遠隔ノードへの転送のため前記ホスト・システムからアクセスするシリアル・バス・インタフェース。

【請求項７】　請求項６記載のシリアル・バス・インタフェースにおいて、前記遠隔ノードからの前記書き込み要求は前記少なくとも１つのレジスタに記憶のためのアドレス情報とデータ情報の両方を含んでおり、前記ホスト・インタフェースはアドレス・バスとデータ・バスを持つ前記ホスト・バスにこれらアドレス情報とデータ情報の両方を送信するよう動作可能であるシリアル・バス・インタフェース。

【請求項８】　請求項６記載のシリアル・バス・インタフェースにおいて、前記遠隔ノードからの前記読み出し要求は前記少なくとも１つのレジスタに記憶されているアドレスを含んでおり、前記ホスト・インタフェースは、読み出し要求を認識するとアドレス・バスとデータ・バスを持つ前記ホスト・バスに該アドレスを送信するよう動作可能であり、前記データ送信器による前記遠隔ノードへの送信のため該データ・バスからデータを取り出すシリアル・バス・インタフェース。

【請求項９】　シリアル・バスとローカル・ノード上のホスト・システム間をインタフェースするため、および、遠隔ノードにより前記シリアル・バス上に置かれた情報を前記シリアル・バスから受信しかつ該受信した情報を前記ホスト・システムに転送し、また、前記ホスト・システムから情報を受信しかつ該受信した情報を前記遠隔ノードによる受信のため前記シリアル・バスに転送するための方法であって、

前記遠隔ノードにより生成されたデータを前記シリアル・バスからデータ受信器で受信するステップと、

データを前記遠隔ノードでの受信のため前記シリアル・バスに送信するステップと、

少なくとも１つのレジスタがそこに受信データを記憶のため前記遠隔ノードによりアドレス可能である複数のレジスタを設けるステップを含み、前記受信ステップは読み出し動

作の間受信したデータを前記少なくとも１つのレジスタに記憶するよう動作可能であり、また、前記送信ステップは書き込み動作の間データを前記シリアル・バスに送信するようになっており、さらに、前記ホスト・システム上のホスト・バスに直接インタフェースするためのホスト・バス・インタフェースが設けられていて、該ホスト・バス・インタフェースは、前記少なくとも１つのレジスタに記憶されたデータを、書き込み動作の間、該データが受信されて前記少なくとも１つのレジスタに記憶されるとき、前記ホスト・バスに転送し、また、読み出し動作の間データを前記ホスト・バスから取り出すようになっている方法。

【請求項１０】　請求項９記載の方法において、シリアル・バス・インタフェースが標準レジスタ空間を含んでいて、前記少なくとも１つのレジスタは該標準レジスタ空間の一部を占有している方法。

【請求項１１】　請求項９記載の方法において、前記複数のレジスタのうちの選ばれたレジスタは標準バス・インタフェース情報の記憶に専用に供されていて、該標準バス・インタフェース情報に関連付けられた前記複数のレジスタを遠隔ノードによって直接アドレスしてそこから該情報にアクセスするステップをさらに含み、前記受信ステップは前記複数のレジスタのうちの１つへのアクセス要求を認識するよう動作可能であり、前記送信ステップはアドレスされるときその内容を送信するよう動作可能である方法。

【請求項１２】　請求項１１記載の方法において、前記複数のレジスタのうちの選ばれたレジスタはコンフィギュレーション・レジスタから成っていて、該コンフィギュレーション・レジスタは前記シリアル・バス・インタフェースの動作を定めるコンフィギュレーション情報のために用いられ、それにより、遠隔ノードは前記コンフィギュレーション・レジスタの１つにアクセスすることによって前記シリアル・バス・インタフェースの動作をプログラムできるようになっている方法。

【請求項１３】　請求項９記載の方法において、前記受信ステップで受信されるデータおよび前記送信ステップで送信されるデータはデータ・パケットであり、該データ・パケットは前記シリアル・バス上の送信ノードを識別するのに必要な情報と、該データ・パケットの内容と、データ・パケットを受信するよう指定された遠隔ノードを識別する情報を含んでいる方法。

【請求項１４】　請求項１３記載の方法において、各データ受信または送信動作に先行して前記遠隔ノードからの書き込み要求または読み出し要求のデータ要求があり、該要求は前記受信されたデータ・パケット内に含まれており、書き込み要求に関連した前記受信データ・パケットは前記少なくとも１つのレジスタに記憶されたそれに関連したデータを含んでいて、前記ホスト・インタフェースはこの書き込み要求を認識して前記少なくとも１つのレジスタに記憶された前記データを前記ホスト・バスに転送し、読み出し要求では、前記ホスト・インタフェースはこの読み出し要求を認識して該データに前記送信ステップでの遠隔ノードへの転送のため前記ホスト・システムからアクセスする方法。

【請求項１５】　請求項１４記載の方法において、前記遠隔ノードからの前記書き込み要求は前記少なくとも１つのレジスタに記憶のためのアドレス情報とデータ情報の両方を含んでおり、前記ホスト・インタフェースはアドレス・バスとデータ・バスを持つ前記ホスト・バスにこれらアドレス情報とデータ情報の両方を送信するよう動作可能である方法。

【請求項１６】　請求項１４記載の方法において、前記遠隔ノードからの前記読み出し要求は前記少なくとも１つのレジスタに記憶されているアドレスを含んでおり、前記ホスト・インタフェースは、読み出し要求を認識するとアドレス・バスとデータ・バスを持つ前記ホスト・バスに該アドレスを送信するよう動作可能であり、前記データ送信器による前記遠隔ノードへの送信のため該データ・バスからデータを取り出す方法。